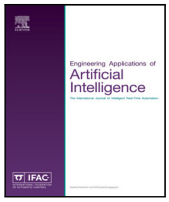
Engineering Applications of Artificial Intelligence 134 (2024) 108631

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research paper

Wheel odometry model calibration with neural network-based weighting ☆

Máté Fazekas [a,b,*], Péter Gáspár [a]

[a] HUN-REN Institute for Computer Science and Control, Hungarian Research Network (HUN-REN SZTAKI), Kende u. 13-17., Budapest, 1111, Hungary
[b] Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics (BME-KJIT), Stoczek utca 2., Budapest, 1111, Hungary

ARTICLE INFO

Keywords:
Wheel odometry
Parameter identification
Neural network
Online self-calibration
Localization

ABSTRACT

The online self-calibration is a required capability from an autonomous vehicle that should operate lifelong in a safe manner. This paper introduces relative weighting by a neural network into the batch Gauss–Newton calibration method of the wheel odometry model. A wheel odometry model with accurately estimated parameters could improve the motion estimation task of an autonomous vehicle, but the online parameter identification from only onboard measurements is a challenge due to the noises and the nonlinear behavior of the dynamic system. A possible solution to deal with the effect of noises is to calibrate the model with more segments at once forming a batch, but this can only reduce the distortion effect, not eliminate it. Our proposed algorithm improves this batch formulation by integrating relative weights for the segments to mitigate the distorting effect of noisy measurements. The method applies an AI-based tool to extract a proper weighting strategy from the previously recorded data that utilizes only online signals during operation. With the usage of the proposed architecture, the calibration accuracy significantly increased with the reduction of the distortion effect of faulty measurements, while the same amount of data is used as the raw batch estimation. The performance of the method is demonstrated with real measurements in a city driving with a passenger vehicle, where the calibration signals come from the equipped automotive-grade type of Global Navigation Satellite System and Inertial Measurement Unit.

1. Introduction

1.1. Wheel odometry in the motion estimation of robots

Nowadays, autonomous robots are becoming widespread both in small-sized cases, *e.g.* mobile robots in warehouses, and in real-sized cases with passenger cars as self-driving vehicles. Fully autonomous behavior can only be achieved with accurate and simultaneously robust state estimation, the most important signals are the velocities and pose (position and orientation). These can be estimated with a wide range of sensors, such as GNSS (Global Navigation Satellite System), IMU (inertial measurement unit), encoder-based wheel odometry, and perception sensors, *e.g.* LiDAR or camera. In the industry, cost-efficiency is also important, thus low-cost automotive-grade types of sensors are generally equipped.

A detailed survey of the odometry-based navigation can be found in Mohamed et al. (2019). The disadvantage of the GNSS is the low frequency and the reduced accuracy in crowded urban environments due to the multipath error and building occlusion (Gao et al., 2018; Falco et al., 2017). The IMU is an error-prone sensor, the bias and noise result in an intolerable signal-to-noise ratio in vehicle applications with a lack of high accelerations (Thrun et al., 2006; Funk et al., 2017). The camera-based estimation is only accurate when enough features are detected and often requires prior knowledge about the motion (Funk et al., 2017; Scaramuzza and Fraundorfer, 2011). These drawbacks can be mitigated with the integration of the signals of the wheel encoder (Falco et al., 2017; Thrun et al., 2006; Funk et al., 2017), a sensor that is equipped on wheeled robots and vehicles anyway. Moreover, in indoor applications (depots, stores, homes, or a mine), the usage of wheel odometry for localization is unavoidable. Furthermore, an accurate wheel odometry model would open up new possibilities, such as pseudo-range measurement correction of GNSS modules, bias compensation of IMU, or scale factor estimation in visual odometry.

☆ The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems (RRF-2.3.1-21-2022-00002). The research was also supported by the National Research, Development and Innovation Office through the project "Cooperative emergency trajectory design for connected autonomous vehicles" NKFIH: 2019-2.1.12-TÉT_VN. The work of Máté Fazekas was supported by the ÚNKP-23-3-II-BME-156 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

* Corresponding author.
E-mail addresses: mate.fazekas@sztaki.hun-ren.hu (M. Fazekas), peter.gaspar@sztaki.hun-ren.hu (P. Gáspár).

https://doi.org/10.1016/j.engappai.2024.108631
Received 7 October 2023; Received in revised form 7 March 2024; Accepted 14 May 2024
Available online 31 May 2024

One limitation of wheel odometry is the possible slip (Mohamed et al., 2019), but this appears more in small robots and uneven terrain and is less significant for self-driving cars. The core drawback is the drift resulting from the parameter uncertainty, which is more significant in the case of a passenger car because an improved wheel model has to be used to capture the varying rolling radius, thus self-calibration is required. Since the parameter values of the wheels are not constant over the life cycle of the vehicle, *e.g.* due to tire wear or swap between summer/winter types, and could change even daily because of the vary in pressure or load, automatic online calibration using the signals of the onboard sensors is a required capability from an autonomous vehicle.

1.2. Related works of wheel odometry calibration

Since the wheel odometry calibration task is basically the same for indoor small mobile robots and passenger vehicles, vehicle types are not distinguished when reviewing the methods. The first methods in Borenstein and Feng (1996) and Lemmer et al. (2010) operate with pre-defined special straight and circular paths for calibration. The error sources are separated into systematic and non-systematic ones, and estimation methods using the final position in Lee et al. (2010), or orientation in Jung et al. (2016) are developed. This type of odometry error approach is generalized to any path in the work of Martinelli (2002) with a covariance matrix, which depends on 4 parameters (2-2 systematic and non-systematic) and the path followed by the mobile robot. Besides special paths, these algorithms often require precise pose measurements, *e.g.* with an expensive DGPS sensor in Lemmer et al. (2010), external vision system in Lee et al. (2010), or human intervention in Martinelli (2002).

Another way to handle the accumulation of odometry errors is to assume an additive component in the velocity inputs of the model. In Roy and Thrun (1999), the deviations are modeled as linear functions of the traveled distance. In this formulation, the model calibration implies estimating these slope values of the linear functions. The odometry compensation uses an onboard LiDAR sensor, and the estimation is based on the maximum likelihood method utilizing the scan matching of two consecutive laser measurements. This results in a highly varying estimation, therefore exponential smoothing is also applied. It can be an advantage to adapt rapidly to the changes in the odometry error, but the tuning of the smoothing as a balance between the adaption and mitigation of noise is an open question. This type of error modeling is examined in detail in the work of Kelly (2004), where a linearization technique based on the first-order behavior of the nonlinear dynamics is presented. The Jacobians depend on the state, input, and also on trajectory. The aims of this linear error propagation model are to determine optimal error compensation approaches and accentuate or attenuate response to individual error sources. Nevertheless, the error due to parameter uncertainty in odometry models, especially for car-like vehicles, can be much more complex than an input linear error function.

Due to the dynamic relation between the vehicle parameters and the pose measurements utilized for their estimation, the calibration task can also be formed as the augmentation of a filtering problem. This method is applied in Larsen et al. (1998) as an Augmented Kalman-filter integrating the model parameters into the state vector and estimating parallel with the physical states. The method widely used for calibration of odometry parameters with several complementary sensors such as, camera (Larsen et al., 1998), gyroscope (Rudolph, 2003), DGPS (Caltabiano et al., 2004), laser range finder (Martinelli et al., 2007), and tested for real-sized passenger vehicles with serial GPS (Brunker et al., 2017). However, in this type of parameter identification, observability issues arise (Martinelli and Siegwart, 2006), the convergence to the true parameter value is not proven (Antonelli and Chiaverini, 2007; Censi et al., 2013), and it is difficult to mitigate the effect of measurement errors, such as outliers or measurement noise (Censi et al., 2013). The advantage of the method is the automatic online operation and low computation demand. Similarly to the input error method in Roy and Thrun (1999), smoothing is necessary, which can be managed with the covariance of the filter, but the proper tuning is challenging.

The general method for parameter estimation is the least squares-based optimization. Antonelli and Chiaverini (2007) forms a linear fitting problem from the nonlinear odometry model in two steps. The model is re-parameterized, and first, only the orientation equation is solved, and its result is utilized in the position equations generating another linear problem. In this way, any bias in the first step is propagated into the second estimation without the possibility of correction (Antonelli et al., 2005). Thus, noise-free orientation measurement is needed, but it is almost impossible in the case of a passenger vehicle in real streets. In the work of Censi et al. (2013), the same first step is applied, followed by a maximum likelihood problem where the location of the laser sensor on the robot is estimated as well. Kümmerle et al. develops a similar method, but the model calibration is integrated into a graph-based simultaneous localization and mapping problem. In Seegmiller et al. (2013), an inverted method is presented where the integration of the linearized system dynamics is calibrated to mitigate the computation effort. Maye et al. (2016) addresses a special problem as the impact of noise on the observability of the location of the sensor and the estimated parameters. With the decomposition of the Fisher-information matrix, it is detected that, due to the noise on the measurements, unobservable directions in the parameter space could appear to be observable.

Although vehicle types have not been distinguished so far, in a real-sized vehicle, effects that are negligible in mobile robots, appear and influence the wheel odometry, such as suspension (Kochem et al., 2002), and steering (Gutiérrez et al., 2007) dynamics, lateral motion (Fazekas et al., 2021a; Brunker et al., 2018), wheel slipping (Brunker et al., 2018) or dynamic load transfer (Fazekas et al., 2020). With the application of a more complex odometry model, the simplifications, such as (Censi et al., 2013; Antonelli et al., 2005), or (Seegmiller et al., 2013) cannot be applied.

From the identification scope, the noise analysis reveals new issues from the theoretical point of view in the case of nonlinear models compared to linear ones. Most methods in the estimation field, such as least squares (LS) or Kalman-filtering assume white noises. However, as the input or process noise is propagated through the nonlinearity, the Gaussian assumption is no longer valid on the output (Ljung, 2010a). Due to the required linearization in the LS solution (Tangirala, 2015), the imprecise initialization has the same consequence (Ljung, 2010b). Accordingly, even with white measurement noises, unbiased model calibration is inherently questionable (Schoukens and Ljung, 2019). Nevertheless, with the spread of automated vehicles, the usable computation capacity and data for the calibration algorithms is significantly increased (CleanTechnica, 2019), thus new methods, *e.g.* the application of machine learning techniques, can be and should be developed.

One of the first papers that apply machine learning techniques in the odometry calibration topic is (Xu and Collins, 2009), where a neural network is trained to learn the pose error of a mobile robot. The method is an offline calibration where the output pose measurement is captured by an external LiDAR. Similar error modeling is presented in Onyekpe et al. (2021) and an improved version in He et al. (2023) for real-sized vehicles, but the online recalibration is not addressed. Wide range of neural network types is tested for end-to-end learning, such as: Recurrent Neural Network in Onyekpe et al. (2021), Transformer in He et al. (2023), Long Short Term Memory in Fariña et al. (2023), and Residual Reduction Modules in Navone et al. (2023). This type of learning method is often supplemented with other sensors, *e.g.* IMU or special gyro (Brossard and Bonnabel, 2019). Other works, such as (Toledo et al., 2018) and Zhang et al. (2021), approximate the whole odometry model instead of the error.

Regardless of whether the error or the model is learned, special attention should be taken to avoid overfitting as the training data's actual measurement error includes the noise and it is also approximated. Another disadvantage is that the industry prefers physical models to black-box models, especially for safety-critical systems such as automated road vehicles. Furthermore, these algorithms often operate with signals from expensive sensors, *e.g.* fiber optic gyro (Brossard and Bonnabel, 2019), or laser-based wheel radius sensor (Toledo et al., 2018) to minimize measurement noise.

Nevertheless, the authors also see potential in these types of learning, but rather as a supplementary method and consider more important the detailed physical modeling of the system dynamics and the calibration of its parameters, before the approximation of the remaining error terms. Therefore, this paper focuses on parameter identification and the development of an algorithm that also addresses online re-calibration.

1.3. Proposed approach

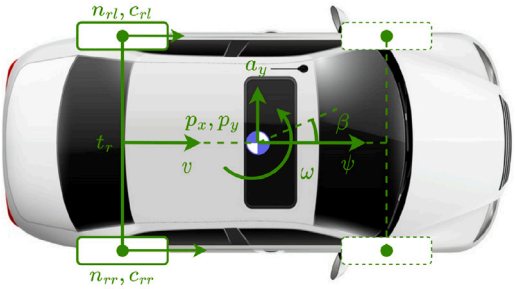
In our previous works, compensation methods for the input wheel rotation noise in Fazekas et al. (2022), and output pose measurement noise in Fazekas et al. (2021b), have been developed to improve the estimation accuracy. However, these are separate algorithms and can be applied only offline. Nonetheless, these papers demonstrate that batch formulation with more measurement segments can be an option to reduce the impact of noise, but weighting is recommended for various segments. It is also mentioned in Seegmiller et al. (2013) but has not been addressed in any related works.

This paper presents our idea to deal with the distortion effect of all the noises. The assumption is that the odometry model can be calibrated well from a batch of around a dozen segments if the impact of the few segments that are accountable for the bias is mitigated. This is achieved by introducing relative weights for each segment that are determined with a neural network. In the design of the algorithm, special attention has been taken to utilize only signals from currently available segments directly for the parameter estimation. In this way, the model can be re-calibrated online only from signals of onboard automotive grade-type sensors. Before going into the details, we would like to motivate the effort by outlining the novelties of the presented approach:

- To the best of the authors' knowledge, the idea of improving the estimation performance with the integration of relative weights in such a calibration task is unique.
- A full calibration architecture is proposed for the online self-calibration of the wheel odometry model of an autonomous vehicle. The algorithm is based on the raw batch Gauss–Newton method completed with an integrated relative weight estimation by a neural network, but only the same signals are utilized.
- Since these weights are only inner, immeasurable, and ambiguous, a proper method that can generate the required label values for the training is developed

Although our work deals with passenger cars, the proposed method can also be used for wheel odometry calibration of any kind of robot.

The paper starts with presenting the wheel odometry model and examining the parameter sensitivity and calibration in Section 2. Then, the Gauss–Newton-based parameter identification is summarized, including the calibration issues to handle. At the end of this Section 3, the idea of integrating relative weighting into the identification algorithm is presented as well. The development of the weight estimator neural network can be found in Section 4, focusing on the formulation of the inputs and label generation in detail. At the end of this part, a block diagram illustrates the full calibration architecture. Section 5 shows the experimental data on which the method is validated. Section 6 contains the results of the model calibration and validation of the proposed method in 4 parts, and a brief evaluation and comparison of the localization with the calibrated odometry model is demonstrated as well. Finally, the paper is concluded in Section 7.



p_x, p_y, ψ : positions and orientation

v, ω : longitudinal and angular velocity

n_{rl}, n_{rr} : rear left/right wheel rotation

c_{rl}, c_{rr} : rear left/right wheel circumference

a_y, β : lateral acceleration, sideslip angle

Fig. 1. Two-wheel odometry model.

2. Vehicle model and motivation

2.1. Two-wheel odometry model

Odometry is the process of using motion measurements to calculate the change of position $p_{x,t}, p_{y,t}$ and orientation ψ_t. Since the lateral dynamic is significant in a real-sized vehicle, the sideslip angle β_t is also integrated into the model.

Between the t_s sampling time, moving on an arc of a circle is assumed, the discrete form of the state transition equations are,

$$\begin{bmatrix} p_{x,t+1} \\ p_{y,t+1} \\ \psi_{t+1} \end{bmatrix} = \begin{bmatrix} p_{x,t} + v_t \cdot t_s \cdot \cos(\psi_t + \omega_t/2 \cdot t_s + \beta_t) \\ p_{y,t} + v_t \cdot t_s \cdot \sin(\psi_t + \omega_t/2 \cdot t_s + \beta_t) \\ \psi_t + \omega_t \cdot t_s \end{bmatrix}. \tag{1}$$

The motion estimation is controlled by the v_t longitudinal and ω_t angular velocity, which are calculated by utilizing the rotation of the wheels in the case of wheel odometry. We apply the two-wheel model illustrated in Fig. 1, where the velocities are based only on the $n_{rl,t}, n_{rr,t}$ rotation of the rear wheels,

$$v_t = (n_{rl,t} \cdot c_{rl,t} + n_{rr,t} \cdot c_{rr,t})/2, \tag{2a}$$

$$\omega_t = (n_{rr,t} \cdot c_{rr,t} - n_{rl,t} \cdot c_{rl,t})/t_r, \tag{2b}$$

where t_r is the rear track width and c_{rl} and c_{rr} are the wheel circumferences. In almost all related works in wheel odometry, the circumferences are handled as constant parameters, but in the next section, it will be demonstrated that the actual rolling circumferences are required for proper localization thus the slight change due to the vertical load transfer is integrated,

$$c_{rl,t} = c_e + c_d/2 + d \cdot a_{y,t}, \tag{3a}$$

$$c_{rr,t} = c_e - c_d/2 - d \cdot a_{y,t}, \tag{3b}$$

where $a_{y,t}$ is the lateral acceleration measured by the IMU. Therefore, the wheel circumferences are now variables and parameterized by the c_e effective wheel circumference, c_d difference between the effective values, and the d dynamic component models the impact of vertical dynamics.

The $p_{x,t}, p_{y,t}, \psi_t$ pose state variables are also measured ones, thus the summarized system model for the parameter identification task is,

$$x_{t+1} = f(x_t, u_t, \theta), \quad x_t = [p_{x,t}, p_{y,t}, \psi_t]^T, \quad y_t = x_t, \tag{4}$$

where $f(\cdot)$ contains (1). The inputs are

$$u_t = [n_{rl,t}, n_{rr,t}, \beta_t, a_{y,t}]^T, \tag{5}$$

Table 1
Values and localization errors with the nominal and reference parameter values.

		c_e [m]	c_d [mm]	t_r [m]	d [mm]	E[m]
Nominal		1.98	0	1.545	0	–
Reference		1.949	2.223	1.518	0.827	–
CASE	c_e nom	$c_{e,nom}$	$c_{d,ref}$	t_{ref}	d_{ref}	1.61
	c_d nom	$c_{e,ref}$	$c_{d,nom}$	t_{ref}	d_{ref}	18.9
	t_r nom	$c_{e,ref}$	$c_{d,ref}$	t_{nom}	d_{ref}	0.67
	d nom	$c_{e,ref}$	$c_{d,ref}$	t_{ref}	d_{nom}	3.45
	all nom	$c_{e,nom}$	$c_{d,nom}$	t_{nom}	d_{nom}	22.3

of which the $n_{rl,t}, n_{rr,t}$ are measured with the wheel encoders, β_t and $a_{y,t}$ are filtered from GNSS and IMU measurements. The parameters of the system to be estimated are grouped to the $\theta = [c_e, c_d, t_r, d]$ vector.

2.2. Parameter sensitivity and requirement of calibration

The disadvantage of this type of localization method is the sensitivity to parameter uncertainty, which corrupts the velocity measurements that are integrated over time to give pose estimates. Since the wheel and track parameters are geometry ones, nominal values can be reached in the vehicle's datasheet. Analysis with our test vehicle, where the localization error using the nominal and reference (which result in the minimum error considering the whole measurement, Section 5.4) values of the parameters, and the error sensitivity are examined on a real 30 km long experiment in Budapest.

The parameter values can be found in the first three rows of Table 1, and in the furthers, 4 test cases are presented in which one of the parameters is set to the nominal values, and for the rest the reference ones are used. In the last row, the model is tested with only nominal values.

The model settings are tested on 300 m long segments and the mean localization error can be found in the last column of Table 1. Using the nominal value of the parameters one by one, the errors are $1.61, 18.94, 0.67, 3.45$ m for the c_e, c_d, t, d parameters, and this increases to 22.30 m if all parameters are set to their nominal value. Take into account that without calibration, this setting has to be utilized, thus the model calibration is a must.

The other key questions are the parameter sensitivity and the possible change of the values. The sensitivity analysis results are presented in Fig. 2. The position errors due to the uncertainty of the c_e effective circumference and t_r track width are in the few meters range. These parameters vary every time when the tires are changed *e.g.* from winter type to summer, and the size of the tire decreases continuously due to wear. Thus, even if nominal values are available for these parameters, the few *cm* deviations from the actual value have to be estimated repeatedly.

The impact of the c_d circumference difference and d dynamic constant are an order of magnitude higher, only a few mm deviations from the optimal value can cause around 20 m localization error, and nominal values are not available at all. Moreover, this tiny difference in the circumference of the wheels changes frequently, depending on the current tire pressure or load of the vehicle *e.g.* the number and weight of the passengers. Therefore, frequent identification of these parameters is required.

In summary, the wheel odometry-based estimation can be integrated into the state estimation layer of an autonomous vehicle only with a proper calibration algorithm, that is able to estimate the parameters from onboard measured signals (without any application engineering), and the process has to be online since there are parameters with significant impact depending on actual circumstances.

3. Model calibration algorithm

3.1. Parameter estimation of nonlinear dynamic models with Gauss–Newton method

Generally, the parameter estimation is formulated as a least squares optimization problem, to minimize the error of the $\hat{y}_k(\theta)$ predictor of the model from the $\tilde{y}_k$ output measurements, such as

$$\hat{\theta}_{opt} = \arg\min_{\theta} V(\theta) = \arg\min_{\theta} \sum_{k=1}^{N_k} \|w_x(\tilde{y}_k - \hat{y}_k(\theta))\|^2, \tag{6}$$

where the predictor contains the system model of (4),

$$\hat{y}_k(\theta) = f(\hat{x}_k, \tilde{u}_k, \theta). \tag{7}$$

When the model is nonlinear in θ, the optimization can only be solved with numerical search (Tangirala, 2015). We apply the Gauss–Newton (Tangirala, 2015) method that solves the nonlinear least squares problem with Taylor-approximation in the following way,

$$\hat{y}_k(\theta) \approx \hat{y}_k(\hat{\theta}_{i-1}) + \underbrace{\left.\frac{\partial \hat{y}_k(\theta)}{\partial \theta}\right|_{\hat{\theta}_{i-1}}}_{j_k} \underbrace{(\hat{\theta} - \hat{\theta}_{i-1})}_{\widehat{\Delta\theta}}, \tag{8}$$

where due to the dynamic behavior of the predictor, the j_k jacobians are computed recursively. This results in a locally linear LS problem, such as

$$\widehat{\Delta\theta}_{opt} = \arg\min_{\Delta\theta} \sum_{k=1}^{N_k} \|w_x((\tilde{y}_k - \hat{y}_k(\hat{\theta}_{i-1})) - j_k\, \widehat{\Delta\theta})\|^2, \tag{9}$$

which can be solved with the LS solution in an iterative way,

$$\hat{\theta}_i = \hat{\theta}_{i-1} + (J^T W_x J)^{-1} J^T W_x R, \tag{10}$$

where the $J := J(\hat{\theta}_{i-1})$ jacobian and $R := \tilde{Y} - \hat{Y}(\hat{\theta}_{i-1})$ residual matrices are formed from the j_k and $\tilde{y}_k$ and $\hat{y}_k(\hat{\theta}_{i-1})$ values as,

$$\begin{aligned} J(\hat{\theta}_{i-1}) &= \begin{bmatrix} j_1 \\ \vdots \\ j_{N_k} \end{bmatrix}, \quad \tilde{Y} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_{N_k} \end{bmatrix}, \\ \hat{Y}(\hat{\theta}_{i-1}) &= \begin{bmatrix} \hat{y}_1(\hat{\theta}_{i-1}) \\ \vdots \\ \hat{y}_{N_k}(\hat{\theta}_{i-1}) \end{bmatrix} = \begin{bmatrix} f(\hat{x}_0, \tilde{u}_1, \hat{\theta}_{i-1}) \\ \vdots \\ f(\hat{x}_{N_k-1}, \tilde{u}_{N_k}, \hat{\theta}_{i-1}) \end{bmatrix}, \end{aligned} \tag{11}$$

and W_x is a weight matrix. Since the model is linearized around the previous parameters, an initial guess for θ is required,

$$\hat{\theta}_0 = [c_{e,nom}, c_{d,nom}, t_{nom}, d_{nom}]^T \tag{12}$$

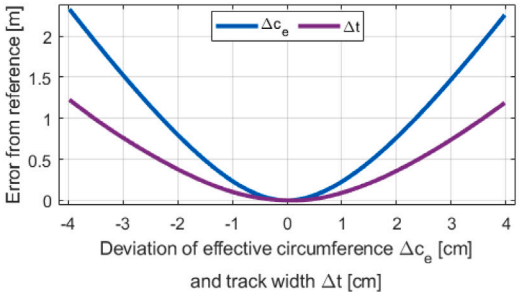
for which the nominal values from the vehicle's datasheet are used. Furthermore, when in the R last term, the $\hat{Y}(\hat{\theta}_{i-1})$ integrated system model is computed with the previous parameters, the states also have to be initialized at the beginning of the estimation window in $\hat{x}_{k=0}$.

3.2. Calibration issues

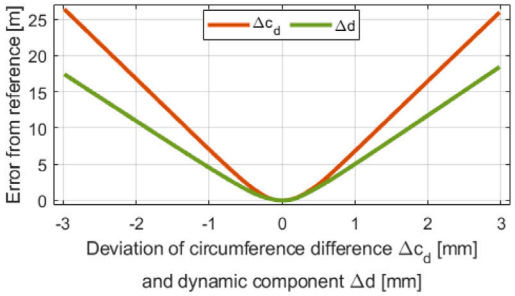
The presented calibration of the nonlinear model is based on the $\{\tilde{u}_t, \tilde{y}_t\}$ input–output signal pairs, thus the main issues arise from the impact of the noises on these measurement signals.

The noise on the measured $\tilde{u}_t$ input can be interpreted as an added unknown process noise on the nonlinear state transition function. Schoukens and Ljung (2019) illustrates that since this distortion enters before the nonlinear transition, using the Gaussian framework to formulate the cost function is no longer realistic. Since the nonlinear least squares methods like GN apply this, the error of the calibration is certainly not zero.

In general, the noise on the $\tilde{y}_t$ measured output would be less significant because it enters after the nonlinearity. However, when the $\hat{y}_k(\hat{\theta}_{i-1})$ predictor is computed in (10), the $\hat{x}_{k=0}$ state has to be initialized



(a) Effective circumference and track width.



(b) Circumference difference and dynamic component.

Fig. 2. Parameter sensitivity of the localization error.

with the associated $\widetilde{y}_t$ measured output. The core issue is that the integrated path of the model utilizing a noisy initial state differs from the ground truth path even with the true parameter values. The impact is similar to the mentioned process noise, consequently, the model calibration can only be biased regardless of the other settings.

Due to these facts, some related works mention that noiseless pose signals are required for the proper calibration however, with a passenger vehicle, in the case of normal city driving, it is almost impossible to produce using only cost-effective sensors, such as ABS encoder, GNSS and IMU. Thus, improvement of the calibration algorithm's side is necessary to ensure bias-free parameter identification.

3.3. Estimation in batch mode

The effect of the mentioned issues can be mitigated if more measurement segments with length N_k are applied at once. In this way, one common θ_B parameter is estimated for the separated segments. The objective function is formulated as,

$$V_B(\theta_B, w) = \sum_{n=1}^{N_n} w_n \underbrace{\sum_{k=1}^{N_k} \|w_x(\widetilde{y}_k - \hat{y}_k(\theta_B))\|^2}_{V_n} \tag{13}$$

where V_n is the sum loss of segment n. Suppose that there are N_n segments ($n = 1 \cdots N_n$), the jacobian and residual matrices of the single GN method can be calculated separately, such as $J_n(\hat{\theta}_{B,i-1})$ and $R_n(\hat{\theta}_{B,i-1})$. In this batch mode, the matrices of the different segments are stacked into the following huge matrices,

$$J_B(\hat{\theta}_{B,i-1}) = \begin{bmatrix} J_1(\hat{\theta}_{B,i-1}) \\ \vdots \\ J_n(\hat{\theta}_{B,i-1}) \\ \vdots \\ J_N(\hat{\theta}_{B,i-1}) \end{bmatrix}, R_B = \begin{bmatrix} \widetilde{Y}_1 - \hat{Y}_1(\hat{\theta}_{B,i-1}) \\ \vdots \\ \widetilde{Y}_n - \hat{Y}_n(\hat{\theta}_{B,i-1}) \\ \vdots \\ \widetilde{Y}_N - \hat{Y}_N(\hat{\theta}_{B,i-1}) \end{bmatrix}, \tag{14}$$

N_n is hereafter referred to as batch size. The parameters can be identified in the same iterative way of (10) with the batch matrices,

$$\hat{\theta}_{B,i} = \hat{\theta}_{B,i-1} + (J_B{}^T W_B J_B)^{-1} J_B{}^T W_B R_B. \tag{15}$$

To avoid confusion, the estimation that is performed on one individual segment (Section 3.1), will be noted as single GN method, while this as batch GN.

3.4. Relative weights to deal with the calibration issues

The batch design of the GN estimation has been introduced with the aim to compensate for these distortions since the model has to fit N_n segments at the same time. However, the formulation can only reduce the bias, because the sum of the individual residuals (R_B) is minimized which leads to the fact that the optimum of the batch is more influenced by the segments where the associated $\|R_n\|$ is higher.

The core idea of our paper is to introduce a w relative weight vector to the batch optimization,

$$w = [w_1 \dots w_n \dots w_N]^T{}_{N_n \times 1}, \tag{16}$$

where the w_n relative weights are added to the objective function of each segment in (13) to mitigate the impact of segments with higher individual residuals resulting from the higher noises. In this way, the W_B batch weight matrix is formed such as,

$$\begin{aligned} W_B &= \mathrm{diag}(w \cdot W_x)_{N_n \cdot 3N_k \times N_n \cdot 3N_k} \\ &= \mathrm{diag}([w_1 W_x \dots w_n W_x \dots w_N W_x])_{N_n \cdot 3N_k \times N_n \cdot 3N_k}, \end{aligned} \tag{17}$$

where the w_n relative weights are scalar numbers. W_x is the same weight matrix for all segments formulated as,

$$W_x = \mathrm{diag}([w_x \dots w_x \dots w_x])_{3N_k \times 3N_k}, \; w_x = [1, 1, 40], \tag{18}$$

and it is responsible for scaling the orientation error since it is measured in rad, consequently significantly lower than the positions measured in m.

Since the w_n weights are not applied because of different physical characteristics of the trajectories, such as lateral acceleration or path length, but because of the distorting effect of the appearing noises, the determination of the w setting that could guarantee the bias-free calibration is a complicated task.

4. Relative weighting with neural network

4.1. Idea of neural network-based weighting

The existence of a weight combination that can significantly improve the model calibration is clear. The main goal of this paper is to develop a method that could determine the proper weights online resulting in improved calibration. The following aspects have been taken into account in the design of the algorithm:

- Each segment has the potential to contribute to an accurate estimation, and our analysis reveals that less good measurements should also be considered but with a lower weight rather than eliminating the segment. Therefore, outlier detection methods, *e.g.* RANSAC are not correspondent.
- The calibration algorithm has to operate online, but previously collected data can be applied to design the method.
- Since the weights are relative (only valid for a given batch), their determination is not explicit and requires a complex algorithm based on non-handmade rules

Considering these, we propose a neural network to determine the weights to take advantage of the enormous data available nowadays in an autonomous vehicle.

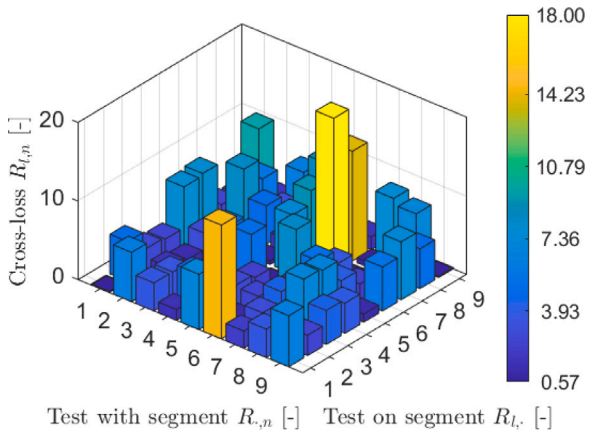


Fig. 3. Cross-losses of the motivation example batch.

4.2. Input for the neural network

The open question is whether it is generally possible to determine the optimal weights resulting in accurate parameter estimation from the available data of the batch. For the batch estimation, measured $\widetilde{u}_k$ inputs and $\widetilde{y}_k$ outputs of the N_n segments of the batch can be used. The dimension of the u and y are 4 and 3 in our model, respectively, thus, *e.g.* with window length $N_k = 1500$, batch size $N_n = 9$, with these are 94,500 values in sum. Using all of these results in a huge network that is difficult to run online on the vehicle and requires enormous data considering automotive applications. Moreover, it is difficult to train as well. Thus, the measured $\widetilde{u}_k, \widetilde{y}_k$ signals are transformed into a lower dimensional space as input of the neural network.

The transformation is based on the recognition that the calibration accuracy can be improved significantly if one can simply guess which segment is "bad" or "good" in relation to the current batch. First, single GN estimations (Section 3.1) are performed for the segments one by one, which result in N_n pieces of $\widehat{\theta}_n$ parameters. Our idea to be implemented is that a segment should be "bad" if its own optimal parameter setting results in a relatively high error on the other segments of the batch or if the optimums of the other segments have a relatively high error on this segment. Therefore, the $\widehat{\theta}_n$ individual optimums are tested on the other segments, and cross errors are computed as,

$$\|R_{l|n}\| = \|\widetilde{Y}_l - \widehat{Y}_l(\widehat{\theta}_n)\| \mid n = 1 \cdots N_n, l = 1 \cdots N_n \tag{19}$$

which is the norm of the residual matrix (11) of segment l with optimal estimated parameters of segment n. Due to the different path lengths of segments, the values are normalized, and the $N_n \times N_n$ cross-loss matrix $\mathcal{R}$ can be formed, *e.g.* Fig. 3 shows the cross-losses of a motivation example batch.

As we can see, the optimal parameters of segments 5, 6, 9 have a higher error on segments 1, 7, 8. Nevertheless, it is not apparent which segments are "good" and "bad", *i.e.* the calibrations of segment 5, 6, 9 are inaccurate, or the segments 1,7,8 are wrong, *e.g.* the noises on the first measurements used for initialization are high. The assumption is that it can be realized from the $N_n \times N_n$ cross-errors with a complex algorithm, *i.e.* a well-trained neural network (Aspect 3 in Section 4.1).

4.3. Label generation for training data

The relative weights are calculated by a neural network whose parameters are determined in a supervised learning method. Thus, training data pairs containing the optimal w_n weights as labels for the $N_n \times N_n$ cross-errors inputs are required. Due to the online calibration, the net inputs have to be based on values available in the actual batch. However, the training of the neural net is performed offline on previously measured data by the autonomous vehicle, thus for label generation more data can be used in addition to the signals of the actual batch. The dataset used in this paper is presented in detail in the section, now we assume that there are N_s measurement segments with measured $\widetilde{u} - \widetilde{y}$ signals containing N_k time instants.

4.3.1. Formulation of label generation task

Since the true values in the θ vector are unavailable, the label values of the w_n weights are determined indirectly. Although the unavailability of optimal θ, the N_s segments can be a proper test set to evaluate a θ parameter vector as,

$$T(\widehat{\theta}) = \sum_{s=1}^{N_s}\left(\sum_{k=1}^{N_k}\sqrt{(\widetilde{p}_{x,k} - \widehat{p}_{x,k}(\widehat{\theta}))^2 + (\widetilde{p}_{y,k} - \widehat{p}_{y,k}(\widehat{\theta}))^2}/N_k\right)/N_s \tag{20}$$

where T is the average test error.

Applying this evaluation metrics, the following objective function is formulated,

$$\overline{w} = \arg\min_{w} T\left(\arg\min_{\theta_B} V_B(\theta_B, w)\right), \tag{21}$$

where in the inner minimization the V_B is the objective function of the batch GN based optimization for θ (13). Therefore, the $\overline{w}$ label weights are the ones that results in a model which has minimal position error on the available test set. The calculation can only be done offline, but the online training of the net is not required.

4.3.2. Optimization of label generation task

The T error is not a direct function of the w weight vector, it can be evaluated with the resulting θ_B parameter after the batch GN estimation is executed with given w_n weights. Thus, the minimization of (21) is solved with genetic algorithm-based optimization because this method does not require any knowledge about the gradient, the pure evaluation of the argument is enough, and an appropriate choice when several local optimums exist.

The genetic algorithm (GA) is a sampling-based optimization method that operates with natural selection inspired by biological evolution. We use the implemented version of GA in MATLAB (Mathworks, 2021), which is based on (Goldberg, 1989). This process is not the topic of the paper, but a brief summary is presented because the appropriate choice of method settings is necessary to properly address the problem we are investigating.

The flow of the process is illustrated in Fig. 4. The method starts with the so-called population generation, the initialization of entities (the optimized parameter) by random sampling. Next, in the fitness calculation phase, every entity is tested with the given objective function and ranked by the loss. The main idea of the GA is to apply genetic operators or functions inspired by the process of natural selection to obtain convergence to the optimum. The *Selection* function chooses the entities as parents to form the new generation of the population. These are paired, and the new entities are calculated with the *Crossover* function. The *Mutation* function enables the GA to search further away from the parent entities' parameter domain, which helps to avoid sticking to a local optimum. Finally, in the *Reinsertion* phase, the new generation is generated from the calculated entities by the previous functions. The stable convergence is ensured with the so-called elite strategy. This iterative evolution of the population continues until the stopping condition is reached.

As we can see, every time a w weight vector entity is evaluated, the batch GN estimation has to be executed. With *e.g.* $N_n = 9$ dimensions, several local optimums could exist, thus many generations and high population size would be necessary for proper convergence, which makes the data generation unmanageable. Therefore, first, the genetic optimization is initialized where the $[0, 1]$ range of every w_n weight is bounded into a smaller one. Thus, the label generation algorithm is formulated in the following three steps.

Step 1: Individual segment validation

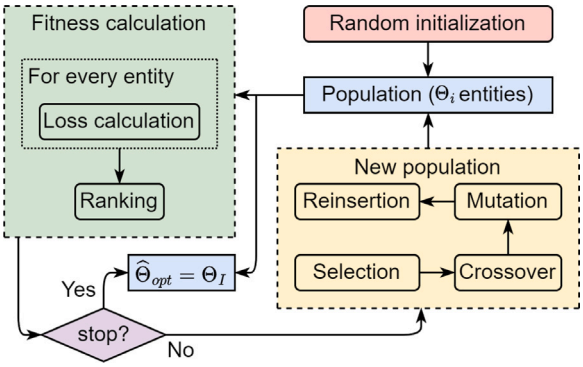


Fig. 4. Architecture of the genetic algorithm-based optimization.

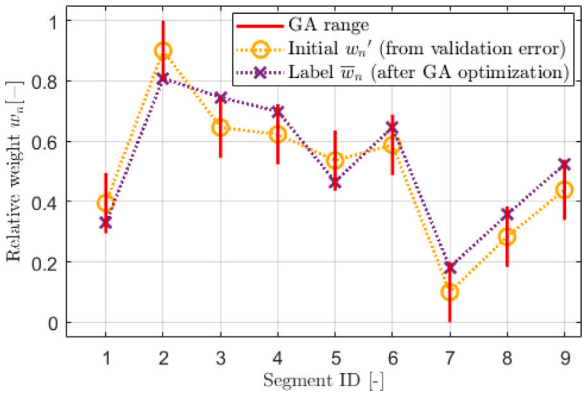


Fig. 5. Label generation for the motivation example batch.

The idea is that the goodness of a segment can be inferred from the error of its resulted parameter if the calculation is based on many segments. For this reason, single GN estimations (Section 3.1) are performed for every segment of the actual batch one by one resulting in $\widehat{\theta}_n$ parameters. These are tested on all the N_s segments of the recorded data,

$$T_n = T(\widehat{\theta}_n) \;\Big|\; \widehat{\theta}_n = \arg\min_{\theta} \sum_{k=1}^{N_k} \|w_x(\widetilde{y}_k - \widehat{y}_k(\theta))\|^2. \tag{22}$$

Step 2: Initialization

These N_n pieces T_n error values are transformed to w_n' initial weight values as,

$$w_n' = 0.8\frac{\max(T_n) - T_n}{\max(T_n) - \min(T_n)} + 0.1, \tag{23}$$

which means that the segment with minimum/maximum error gets 0.9/0.1 values, and the rest are scaled linearly between the two values. These are utilized for lower and upper bounds of the weight values in the entities of the genetic algorithm-based optimization as,

$$w_n' - 0.1 < \overline{w}_n < w_n' + 0.1, \tag{24}$$

Step 3: Weight label calculation

The $\overline{w}$ weight label vector is determined with the genetic optimization. The fitness calculation is based on the introduced objective function in (21). The lower and upper bounds of each segment value in the weight vector entities are set by (24). Thanks to range reduction, the label calculation can be managed with a population size of 3000, over 30 generations.

The results of the label calculation of the presented motivation example batch are illustrated in Fig. 5. The weights of the best entity resulting by the genetic optimization are different from the initial values, *e.g.* the final $\overline{w}_n$ values at segments 2,5 are close to the lower bounds while at $3, 7, 8, 9$ to the upper, respectively, because that weight settings results in more accurate model calibration. This demonstrates that even if such position test errors were available for the segments of the actual batch online, these values would not be enough to determine the appropriate weights for the segments resulting in improved calibration. Consequently, a more complex algorithm, such as a neural network, is necessary.

4.4. Workflow of the calibration algorithm with relative weighting

In this section, the entire calibration process is summarized briefly, and the flowchart of the proposed method is illustrated in Fig. 6. The model calibration is an online process, but it integrates a neural network trained offline from previously saved data by the autonomous car.

The network generation process has 4 phases. The first phase is signal processing, where the raw wheel encoder, GNSS and IMU signals are filtered to form $\mathcal{U}_t$ and $\mathcal{Y}_t$ calibration signals (Section 5.2). The pre-recorded and filtered data is divided into N_s smaller segments, and from these batches are generated with a batch size of $N_n = 9$ (Section 5.3) in the second phase. The training data is created in the third phase, where the input of the neural net and the labels are computed for every generated batch (Sections 4.2 and 4.3). This process results in N_b pieces of $\mathcal{R} - \overline{w}$ training data pairs. Based on these pairs, the neural network training is carried out with the backpropagation algorithm in phase four (Section 6.1).

The bottom light green part of Fig. 6 illustrates the block diagram of the online model calibration process. The signal filtering and input calculation are the same, and the offline trained neural network is utilized to estimate the w_n weights for the actual batch. These are integrated into the formulation of the W_B weight matrix, and the θ vehicle parameters are obtained by the batch GN method (Section 3.3).

5. Measurement data for the odometry calibration

5.1. Measurement scenario

The proposed method is tested with data saved with a real series vehicle that is equipped with automotive-grade dual-GNSS and IMU sensors. The last contains a MEMS-based 3-axis accelerometer, gyro, and compass. The wheel rotation measurements come from the onboard ABS encoders via the CAN bus of the vehicle. The signal sampling frequency in the calibration process is chosen to 40 Hz, thus the odometry model is discretized with $t_s = 0.025$ s sampling time.

It has been mentioned that the method is developed to calibrate the vehicle model online in everyday operations using onboard signals. Thus, the used measurement data consists of a 30 km long driving in normal traffic conditions. The route is in the suburb and city area in the southwest of Budapest, the path can be found in Fig. 7. The track contains various bends with low and high speeds as well and lots of crossroads.

5.2. Signal filtering

From the inputs of the odometry model, the n_{rl}, n_{rr} and a_y signals are measured with the ABS encoders and the IMU, respectively. The β sideslip angle is included because neglecting the lateral motion corrupts the parameter estimation (Fazekas et al., 2021a). The quantity cannot be measured directly in series vehicles, but its estimation is a well-explored topic. Our implementation utilizes the GNSS and IMU signals in a sensor fusion method similar to Bevly et al. (2006).

The parameter identification requires measurements of the p_x, p_y position, and ψ orientation outputs of the model. These can be measured directly with the dual GNSS, but the accuracy is poor. Since the signals of the GNSS and compass are assumed to be noisy but unbiased, and the

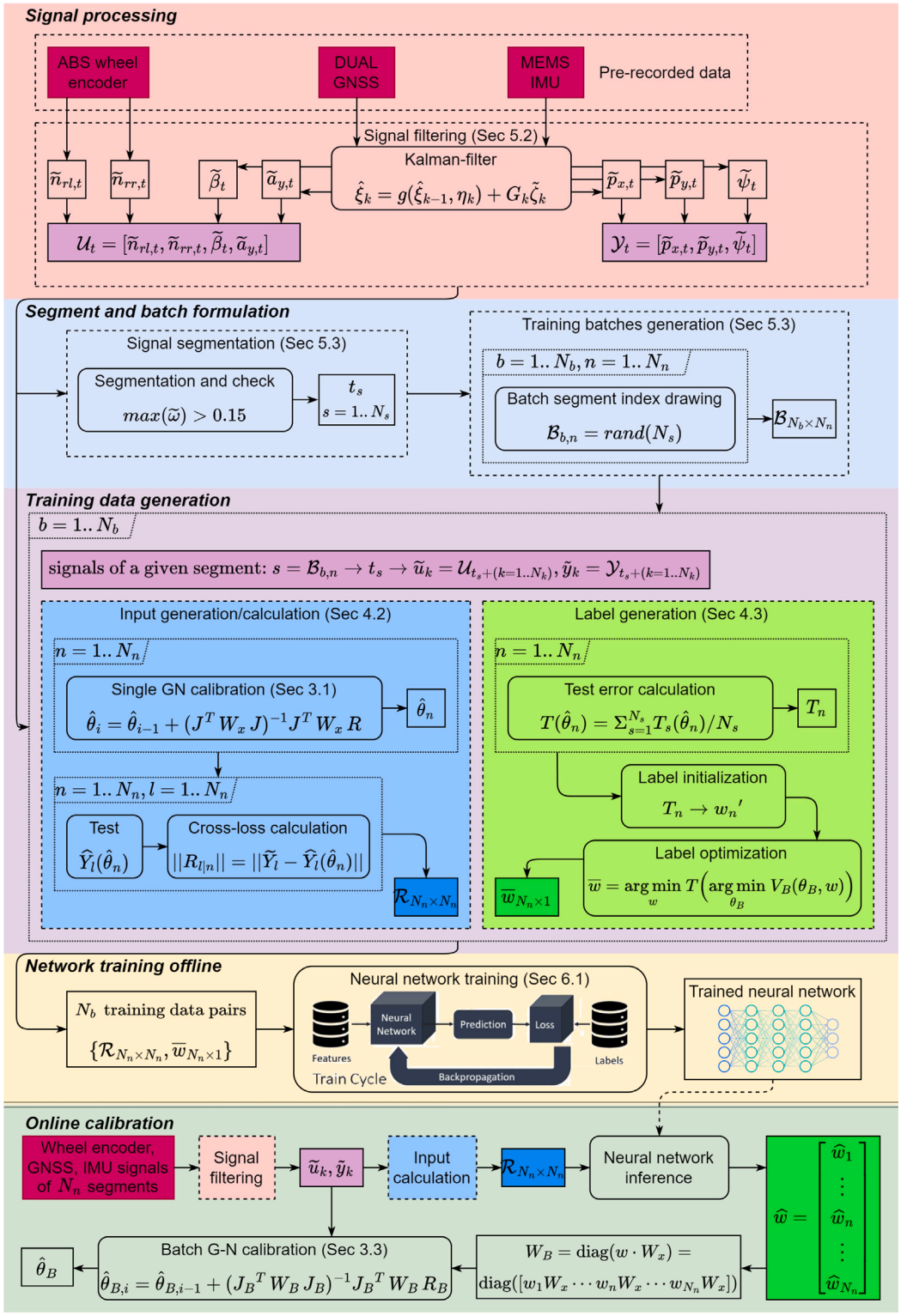


Fig. 6. Architecture of odometry model calibration with neural network-based relative weighting. The first four phase illustrates the offline training of the weight estimator neural net on pre-recorded data. The last green box presents the online calibration steps with the actual onboard signals where the pre-trained network is used in inference mode.

acceleration and gyro measurements with the IMU are biased, but the noise is lower, sensor fusion is a preferred solution. This type of filtering has also been developed in detail, thus another Kalman-filter similar to Caron et al. (2006) is implemented to obtain the pose measurements.

The model calibration is based on the filtered version of the signals, which are organized into the following variables,

$$\mathcal{U}_t = \left[\tilde{n}_{rl,t}, \tilde{n}_{rr,t}, \tilde{\beta}_t, \tilde{a}_{y,t}\right], \qquad \mathcal{Y}_t = \left[\tilde{p}_{x,t}, \tilde{p}_{y,t}, \tilde{\psi}_t\right]. \tag{25}$$

Fig. 7. Path of the measurement data in Budapest.

5.3. Segment and batch formulation

The calibration architecture operates with smaller measurement parts, therefore the 30 km long route is divided into 300 m average long segments consisting of $N_k = 1500$ time points and 1 s step length between them. The segments when the vehicle should stop are eliminated, and since the c_d, t and d parameters can be only appropriately observed with the yaw rate Eqs. (2b), only the $N_s = 513$ segments with higher absolute angular velocity than 0.15 rad/s are selected for the parameter estimation. The start point of the segments is stored in the variable t_s ($t_s \in t$ where $s = 1 \cdots N_s$) to obtain from the $\mathcal{U}_t, \mathcal{Y}_t$ variables the currently necessary $\widetilde{u}_k, \widetilde{y}_k$ signals for the GN estimations.

The calibration process operates with a batch of $N_n = 9$ segments. Thus, from the 513 individual segments of the measurement data, $N_b = 20000$ batches are formulated to test the proposed method. The data of the batches are organized into the $\mathcal{B}$ matrix with size $N_b \times N_n$, where the rows contain the indexes of the segments of the actual batch. The components of each batch are drawn randomly as,

$$\mathcal{B}_{b,n} = rand(N_s), \quad b = 1 \cdots N_b, n = 1 \cdots N_n, \tag{26}$$

on the condition that the same segment is not drawn twice. For neural network training purposes, it is important that the training data (the batches in our case) are not similar, but for a batch size of 9 out of 513 segments, the chance that two drawn batches contain the same 3 segments is less than 0.0005%.

With this formulation, the calibration signals of a segment in a batch, *e.g.* the ones of the 4th segment of the 3rd batch, can be obtained as,

$$b = 3, n = 4 \rightarrow s = \mathcal{B}_{b,n} \rightarrow t_s \rightarrow \widetilde{u}_k = \mathcal{U}_{t_s+(k=1\cdots N_k)}, \widetilde{y}_k = \mathcal{Y}_{t_s+(k=1\cdots N_k)}. \tag{27}$$

5.4. Validation error

The true values of the parameters are unknown, thus the model calibration could be validated with its output error. The position error is utilized, and since obtaining ground truth measurements in real traffic is difficult, the computation is based on the $\widetilde{p}_x, \widetilde{p}_y$ filtered signals. This evaluation models the GNSS outage scenario, which is a widely used metrics for odometry validation in passenger vehicles (Onyekpe et al., 2021).

This position error is similar to the $T(\theta)$ test error in (20), but here some segments are eliminated. The reachable minimum error is determined offline with a genetic algorithm-based search. An iterative elimination strategy demonstrates that if the last 17 segments with the highest error are eliminated, on the remaining segments, the localization error is lower with around 10%. These wrong segments have position errors higher than 8 m, while the others are in the 1–5 m range. Accordingly, these are definitely outlier measurements and distort the validation of the proposed method.

Of course, these segments are only not taken into account in calculating this validation error, they are applied in the generation of the training batch data. To avoid confusion, the validation error with which the model calibration results are examined in the Results section is denoted by $E(\theta)$,

$$\begin{aligned} E(\theta) &= \sum_{s=1}^{N'_s} E_s(\theta)/N'_s \quad | \\ E_s(\theta) &= \sum_{k=1}^{N_k} \sqrt{(\widetilde{p}_{x,k} - p_{x,k}(\theta))^2 + (\widetilde{p}_{y,k} - p_{y,k}(\theta))^2}/N_k, \end{aligned} \tag{28}$$

where the $N'_s = 496$ refers to the outlier rejection.

Consider that the minimum validation error is not zero independently of the applied calibration method because, for the error computation at the beginning of the segments, the states cannot be initialized without noise (Section 3.2). The reachable minimum error determined with the mentioned genetic algorithm-based search is 2.68 m on the 496 selected 300 m long segments. Therefore, the error of the calibrated models is evaluated from this value in the next section.

6. Experimental results

The inputs of the neural net and the labels are calculated with the algorithms presented in Sections 4.2 and 4.3. The experimental results are presented in 3 phases:

First, the training results of the neural network using the generated labels are presented in Sections 6.1 and 6.2. In a real case, this part is performed offline, *e.g.* in a computation center or in the cloud, and the knowledge from a large amount of historically recorded data is obtained in this part.

Next, in Section 6.4, the proposed calibration algorithm with the integrated neural network-based weighting is illustrated in detail with an example batch with 9 segments. In a real case, this part is performed online on the vehicle using the pre-trained network.

Last, testing the robustness of the algorithm, the results of all of the generated 20,000 batches are presented in Section 6.5, but in this part, only the validation error of the models is examined and summarized.

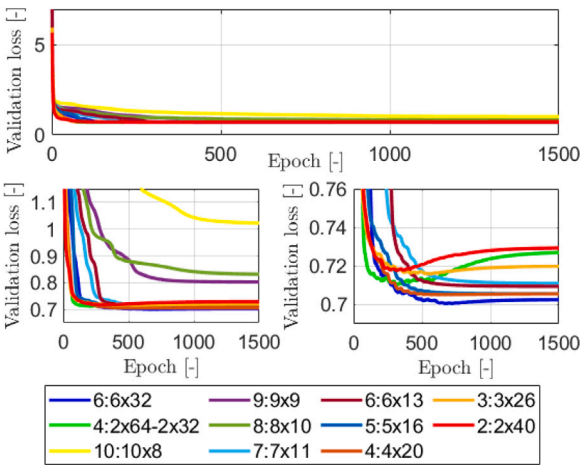


Fig. 8. Validation losses of the various depth neural networks.

6.1. Training of the weight estimator net

In the formulation of the neural network structure, the focus is on the fully connected ones. Convolutional networks have also been tried, but there have been no promising results. The reason could be that the convolutional layers are responsible for extracting features from the raw data. Since cross-errors are appropriate features for deciding the "goodness" of a segment, the raw $\widetilde{u}_k$ input and $\widetilde{y}_k$ output signals of the GN estimation are transformed into the cross-loss matrix $\mathcal{R}$ containing the $N_n \times N_n$ $\|R_{l|n}\|$ errors, thus low-level feature extraction on these does not have much effect. Furthermore, recurrent layers are not considered due to the lack of sequential data.

6.1.1. The required number of neurons

The formulation of the fully connected neural network starts with the determination of the required number of neurons to be able to predict the output correctly. Networks with stacked fully connected hidden layers containing hundreds of neurons in sum are examined. Regardless of the size increase, the validation loss is around 0.7, as the blue line illustrates a network consisting of 6 deep 32 wide (6×32) hidden layers in Fig. 8. This and another example where the 192 hidden neurons are arranged into 4 layers (2×64–2×32) are able to overfit to the dataset clearly, thus, these can be stated as baselines in the context of the examined problem.

Next, networks with a fewer neuron number and the effect of the deepness are studied. Fig. 8 shows 9 other cases where around 80 neurons are stacked uniformly with numbers of layers from 2 to 10. The plot demonstrates that $1/3$ of the neurons of the previous two networks can be enough to reach almost the same performance, the validation errors of the best networks converge to 0.705, only the overfitting does not occur clearly. Furthermore, the deepness of the network has a significant impact, the networks with more than 7 hidden layers cannot achieve the minimum loss, and also the shallow ones with fewer than 4 layers have higher errors. In all of the aforementioned networks, every fully connected layer is followed by a batch normalization and a tangent hyperbolic activation function and trained with various learning parameters to find the optimal settings. In our case, with the input and output layers, *e.g.*, the network with 4×20 hidden layers contains 3089 trainable parameters. Therefore, 80 hidden neurons can be sufficient to predict the 9 outputs from the 81 precalculated cross-error feature input values. Moreover, the validation loss drops from the initial with an order of magnitude. In the impact of depth, two effects should be taken into account. With the same amount of total neurons, the number of trainable parameters increases with wider and shallower shapes: *e.g.*, with 7×11, 5×16, and 3×26 hidden layers, 1802, 2553, and 3779 parameters are available, respectively. However, complex relationships can only be built with a deep network containing more layers since the later layers form a superposed feature of the previous ones. Thus, the 4–6 optimal depth in the context of the examined problem is appropriate.

6.1.2. The optimal neural network shape

The shape of the linked fully connected layers is analyzed as well with 5 different types illustrated in Fig. 9: the previously applied stacked (A), growing (B), and descending (C) pyramidal, autoencoder-like (D) and its counterpart rhombus (E) when the middle layers are the widest.

Only networks with 4 and 6 hidden depths are considered, and the total number of neurons is the same around 80, for proper comparison with the previous ones. The convergence plots of the various network architectures can be found in Fig. 10. The left plots show that the 6 deep slight pyramidals perform worse, but the autoencoder and diamond types can reach the 0.7 loss. It is interesting that with 4 layers, all of the types converge to below 0.7 a bit, and the minimum loss is resulted by the (C) type descending pyramidal shape, which forms the most trainable parameter from the 80 neurons.

In summary, in the examined problem, the outputs can be predicted appropriately with 80 neurons, 4 depths are sufficient to form superposed features from the inputs, and avoiding bottlenecks containing only a few neurons is important. Therefore, the neural network of 4 hidden layers with 32-24-16-8 width is taken forward for final tuning.

6.1.3. Final training of the chosen network architecture

In the chosen fully connected neural network, the width of the 4th hidden layer is increased to 10 to eliminate the possible bottleneck since the output layer has 9 values. The mentioned batch normalization and hyperbolic tangent activations are also included. The detailed architecture of the network can be found in Fig. 11.

With the used batch size of 9 in the GN calibration, this neural network operates with 4095 parameters to estimate the 9 w_n relative weights of the segments from the 9×9 $\|R_{l|n}\|$ cross-losses. The network is trained with the generated 20000 $\mathcal{R}$ cross-loss matrix and $\overline{w}$ label weights input–output pairs by the MATLAB Deep Learning Toolbox (Mathworks, 2022). The data is divided into 80–20% train and validation groups. The values are scaled to $[-1, 1]$ for numeric stability. The solver is the Adam method of Kingma and Ba (2015), where the most relevant learning parameters are,

α_0 : `InitialLearnRate` $[0.2, 0.1, 0.05]$,

$\Delta\alpha$: `LearnRateDrop` $[0.75, 0.80, 0.85]$,

β_1 : `GradientDecay` $[0.8, 0.9, 0.95]$,

β_2 : `SquaredGradientDecay` $[0.9, 0.99, 0.999]$,

where all of the combinations of the 3 values in the ranges are tested to optimize the training. The other parameters of the training are fixed, such as: `LearnRateDropPeriod=50`, `MaxEpochs=2000`, `MiniBatchSize=1600`. Fig. 12 illustrates the validation loss of the 81 parameter setting. The flow of the convergences are different due to the various learning rates, but after 300 epochs, the losses are in the same range, and the minimum value of every case is below 0.68. The minimum loss of the best setting is 0.668 thus, this training is less sensitive to the Adam optimizer parameters in the presented range.

The signal of this best setting is highlighted in red in the figures. This chosen case has the parameters $0.1 - 0.8 - 0.95 - 0.99$ of $\alpha_0 - \Delta\alpha - \beta_1 - \beta_2$, respectively. The validation loss has a minimum of 0.668 at the 766th epoch. The training loss is illustrated as well, it is 0.646 at this epoch, and it decreases only a bit to 0.642 at the end. Thus, the overfitting is negligible, the chosen trained neural network can be the final one as well. The losses drop around an order of magnitude, thus the training of the network parameters converges to a stable optimum.

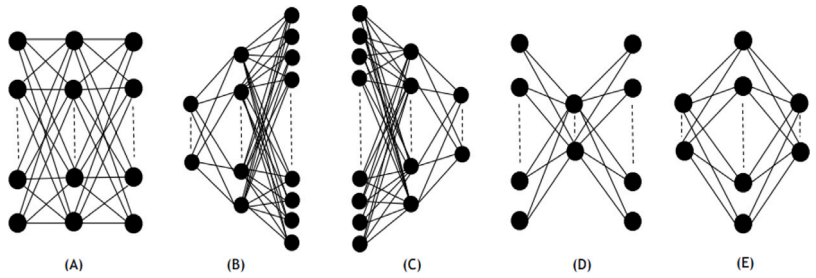


Fig. 9. Tested network shapes in the model architecture analysis (Kulathunga et al., 2021).

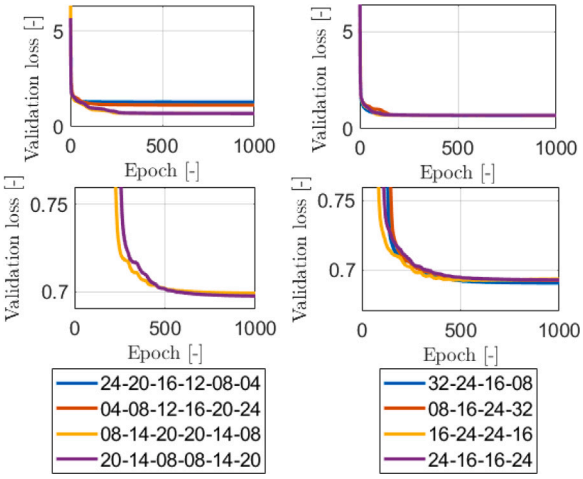


Fig. 10. Convergence plots of the network shape analysis. The values in the labels indicate the width of the layers.

6.2. Results of weight estimation

The previous figure illustrates MATLAB's training loss for the whole data group. Since this applies to the scaled values (only for training) and is uniquely calculated by MATLAB, the $\widehat{w}$ estimated weight vectors are scaled back to the $[0, 1]$ range. Furthermore, because only the relative values between the $\widehat{w}_n$ weight elements in the batch are essential, $sum(\widehat{w}_n) = 4.5$ in each $\widehat{w}$ vector is applied in the re-scaling process.

The overfitting is tested with 5-fold cross-validation, thus the training process of Section 6.1 is performed 5 times each with a different 20% validation group of the data. The estimated weight errors of the batches in the 5 validation folds can be found in Fig. 13.

As we can see, the different training cases have similar errors, therefore overfitting to a smaller part of the data does not occur.

In every case, 75% of the batches have a lower mean error than 0.124, the median is 0.091. The outlier limit with these boxplot settings is 0.205, which is exceeded in 741 cases, but this is only 3.71% of the batch number. The 0.091 median error corresponds to 9.1% regarding the $[0, 1]$ range of the estimated $\widehat{w}_n$ weights. It is difficult to evaluate the performance because the aim is to increase the calibration accuracy, and these weights are only inner variables.

Nevertheless, the element-wise weight error is also calculated and illustrated with a histogram in Fig. 14. Except for some mild peaks around 0.05, the element-wise errors can be exactly modeled with a normal distribution, the fitted function has $3.7765e^{-4}$ center and 0.1337 deviation. Thus, the trained neural network can estimate the label weights without any bias from the chosen input data of the batch.

6.3. Convergence and stability analysis of the GN nonlinear estimation method

In the context of nonlinear least squares methods, the convergence to the global optimum and sticking into a local one is an open question. These are mainly influenced by the proper choice of the $\widehat{\theta}_0$ initial parameter guess (12). Since the calibrated parameters are physical ones, the estimation starts not far from the optimum thus, convergence issues should not occur. Furthermore, a test is also carried out, where the calibration with the example batch is performed with various $\widehat{\theta}_0$ initial parameter settings. The estimation started from three different values for every parameter (c_e:[1.9,1.95,2], c_d:[-5,0,5], t:[1.5,1.55,1.6], d:[-2,0,2]), the final estimated values of all of the 81 combinations can be found in the Appendix in Fig. 24. The standard deviation of the values are lower than 10^{-5} for every parameter, therefore the estimation stably converges to the optimum regardless of the initialization.

Fig. 25 in the Appendix illustrates the path of parameter convergences. In some cases, the estimation of c_e and t starts off in the wrong direction, but this is only the consequence of the negative sign of the c_d and d initial guess. These are only tested to check the robustness of the GN method, the d is formulated as a positive value by default (3), and the c_d difference of the wheel circumstances is initialized to 0 in practice as well. If zero values are applied, the estimations converge to the correct direction straightaway (Fig. 26 in the Appendix).

6.3.1. Comparison with other nonlinear least squares methods

The optimization task and the general form of the nonlinear least squares technique is the following,

$$\arg\min_{\theta} \sum_{k=1}^{N_k} \|\widetilde{y}_k - \widehat{y}_k(\theta)\|^2, \ \widehat{\theta}_i = \widehat{\theta}_{i-1} + \eta_i \xi_i, \tag{29}$$

where ξ_i is the direction of change in the parameter space and η_i is the step length. In the most basic Gradient Descent (GD) technique, the direction is the negative gradient of the V cost function ($\xi_i = -\partial V/\partial\theta$), while the step length is parameterized with a constant $\eta_i = 1/\lambda$. The Newton–Raphson (NR) method operates with the Hessian second derivative of the cost $\eta_i = \partial V/\partial^2\theta$ as step length, and the direction is the same. The utilized Gauss–Newton method is based on the first-order approximation of the $\widehat{y}(\theta)$ predictor (8), which results in the Jacobian of the predictor ($J = \partial\widehat{y}/\partial\theta$) in the two components, while the Levenberg–Marquardt (LM) technique proposes a λI additional damping to the GN one, such as

$$\eta_i = \left(\frac{\partial\widehat{y}}{\partial\theta}\Big|_{\widehat{\theta}_{i-1}}^{T} \frac{\partial\widehat{y}}{\partial\theta}\Big|_{\widehat{\theta}_{i-1}} + \lambda I \right)^{-1}, \quad \xi_i = -\frac{\partial\widehat{y}}{\partial\theta}\Big|_{\widehat{\theta}_{i-1}}^{T} (\widehat{y} - y). \tag{30}$$

The resulted in ξ_i with the predictor approximation is similar to the $-\partial V/\partial\theta$ component, thus the only difference in the 4 approaches is the determination of the step length.

However, the $J^T J$ part in the predictor-based techniques is a suitable approximation of the Hessian around the optimum (Tangirala, 2015), which is ensured in our case due to the physical property

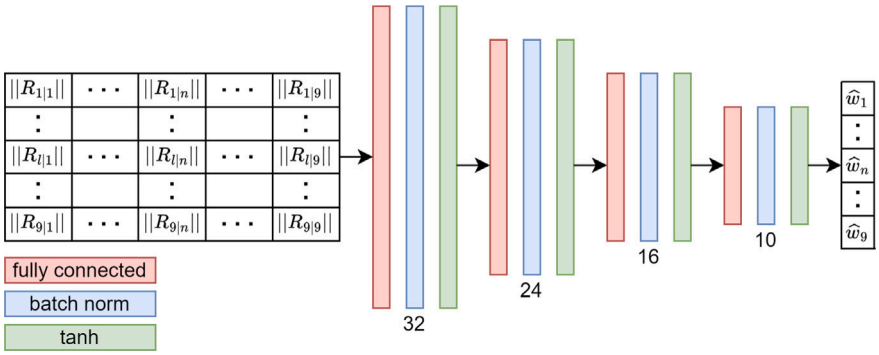


Fig. 11. Architecture of the applied neural network.

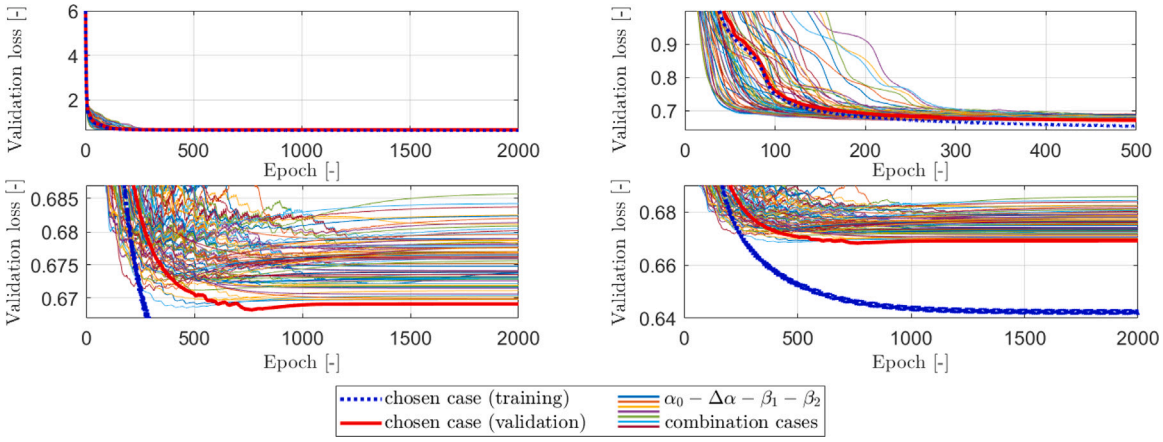


Fig. 12. Losses of the training of the final neural network with the tested 81 Adam optimizer parameter combinations.

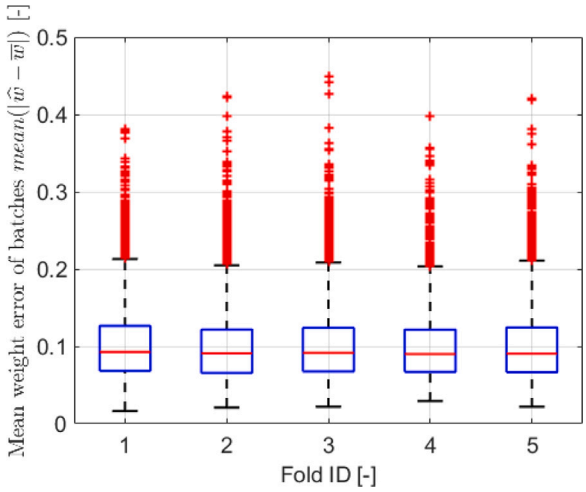


Fig. 13. Results of the 5-fold cross-validation with box plots (whiskers are 1.5 IQR).

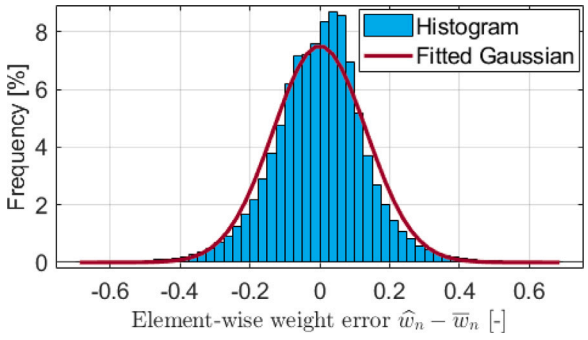


Fig. 14. Element-wise deviation of the weight estimation.

and proper initialization of the parameters. The damping of the LM is mainly required when divergence occurs, moreover, the diagonal elements in the J^TJ matrix are high, since the task is highly overdetermined (the 4×4 matrix is calculated from $N_k = 1500$ time values). Thus, the NR, LM (with low λ), and GN techniques have almost the same output in the examined case.

With increased λ, LM method becomes similar to the GD (Yu and Wilamowski, 2018), since $\lambda I \gg J^TJ \Rightarrow \eta_i = 1/\lambda$. This induces different convergence behavior, therefore the calibration of the example batch is performed with the components of (30) and $\lambda = 10^5$ setting as well. The final parameters are in the same optimum, only the c_e and c_d converge a bit faster, while the t and d are a bit smoother. The plots can be found in Fig. 27 in Appendix. In summary, the utilized method to solve the nonlinear least squares task has less influence due to the proper initialization close to the optimum.

6.3.2. *Convergence behavior in the context of relative weighting*

The convergence behavior of the GN method is also tested in the context of relative weighting. The path of the parameter calibrations can be found in Fig. 15 for the example batch in the raw unweighting, NN-based weighting, and with randomly generated weights. From the last, only 10 cases are illustrated, but 100,000 were performed for exhaustive analysis to visualize the bounds of the estimated parameters final value.

Regardless of the introduced relative weights, the estimation does not diverge in any case. Moreover, the direction of the convergence

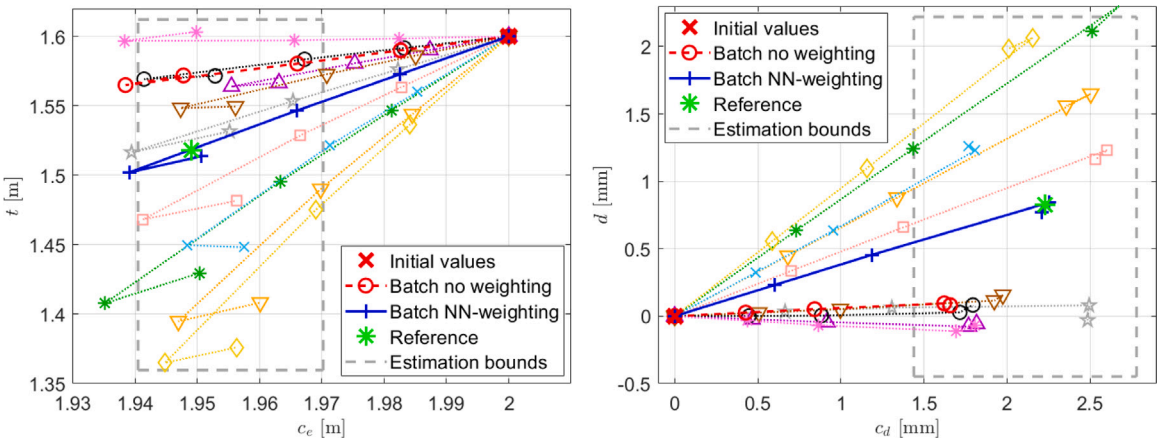


Fig. 15. Parameter convergence plots of the examined batch in cases: without weights, weights calculated by the trained neural network, and with 10 random weight cases.

Table 2
Cross-losses of segment 2 and 7. $R_{l|n}$ means the loss on segment l with estimated vehicle parameters of segment n.

l/n	1	2	3	4	5	6	7	8	9	Mean
$R_{2\|n}$	6.3	0.2	1.6	1.1	2.9	2.5	8.4	5.7	1.8	3.4
$R_{7\|n}$	2.2	3.1	4.3	4.0	5.4	3.6	0.1	0.6	7.7	3.5
$R_{l\|2}$	5.0	0.2	3.5	4.6	2.0	1.1	3.1	1.3	2.6	2.6
$R_{l\|7}$	2.7	8.4	4.9	1.7	9.2	20	0.1	0.7	7.4	6.1

flow is similar, thus the stability of the calibration is not influenced by the weights. However, weighting can substantially modify the optimum location from the no-weighting case, but this phenomenon motivates the whole methodology obviously.

6.4. Calibration of a batch with relative weights

In this section, the model calibration with the proposed batch GN architecture with the integrated neural network-based weighting is demonstrated in detail with an example batch. The used batch is the one that has been illustrated as a motivation example when the relative weighting with the neural network has been examined in Section 4. In this demonstration, the only offline available validation error (Section 5.4) is applied as an evaluation metrics, which is shown in Fig. 18.

The first step of the model calibration is to estimate the $\hat{\theta}_{n=1\cdots N_n}$ optimal vehicle parameters of each segment of the batch individually. The resulting parameters can be found in Fig. 17.

Next, the estimated settings of the segments are tested on the others forming the $R_{l|n}$ cross-loss values. These losses of the best and worst segments, 2 and 7, respectively, are listed in Table 2.

The interesting fact is that when the other segments are tested on these two (the first two rows of the table), the values are substantially different, but the mean values are almost the same, although the segment 7 has 10.11 m validation error compared to the 0.60 m of segment 2. Thus, on a segment, whose estimated optimal parameters are totally wrong (t_r track width is 1.92 m or negative d dynamic tire component), the average loss of 9 other segment calibrations can be the same as on a measurement section that results in almost perfect calibration. This illustrates the difficulty of deciding the goodness of a segment from online data. Nevertheless, if the settings of these two segments are tested on the other ones (the last two rows of the table), it becomes clear that segment 2 is better, but the mean loss of the worse calibration is only 2.3 times higher despite the fact that the rate of the validation errors is 16.8.

The $N_n \times N_n$ $\mathcal{R}$ cross-losses of the example batch have been shown in Fig. 3. This matrix is utilized as input of the offline trained neural network, the output of the net, the estimated $\hat{w}_n$ relative weights can be found in Fig. 16. The estimated values are close to the label ones, furthermore, the speculation at the end of Section 4.2 has also become apparent. It has been obvious that the weight of segments 2, 3, 4 have to be the highest, but the relation between the 1, 7, 8 and 5, 6, 9 segment groups has been an open question at that time. The $\hat{w}_n$ estimated weights declare that low weights should be given to segment 1, 7, 8.

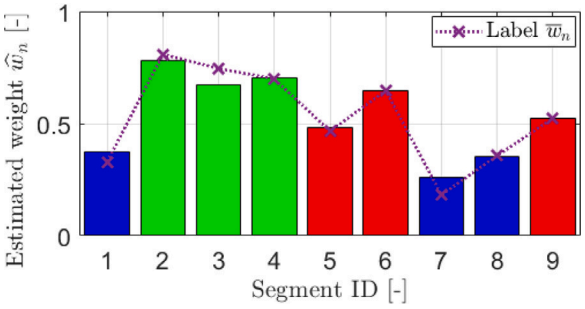


Fig. 16. Estimated weights of the example batch.

If we examine the estimated parameters in Fig. 17, especially the peak values of the resulting t_r track width, the consideration seems to be correct. This is further clarified by the offline calculated validation errors in Fig. 18.

Finally, in the last step of the proposed calibration architecture, the estimated $\hat{w}_n$ relative weights are utilized in the W_B weight matrix of the batch GN calibration. The outcomes of this estimation can be found in the parameter and error figures as well, in parallel with the results of the batch calibration without relative weighting.

The formulation of a raw batch problem from the individual segments significantly improves the calibration performance because the model of the "batch no weighting" case has 2.42 m validation error, lower than the second best of the individual ones whose mean error is 5.34 m with a standard deviation of 2.74 m. Nevertheless, this calibration is still biased, *e.g.* the estimated c_d, which is the most sensitive vehicle parameter, has a higher error than most of the segment ones. The most likely explanation for this is that segment 9 has a peak low c_d value, and since this segment is considered with the same importance, the common batch estimate cannot deviate significantly from this.

However, if one can properly guess the w_n relative weights (Fig. 16), the validation error decreases to a negligible low 0.13 m as it is illustrated with the "batch NN-weighting" case in Fig. 18. This also means that the calibration is almost bias-free, the error of the estimated vehicle parameters are c_e : -2.91 mm, c_d : 0.001 mm, t :

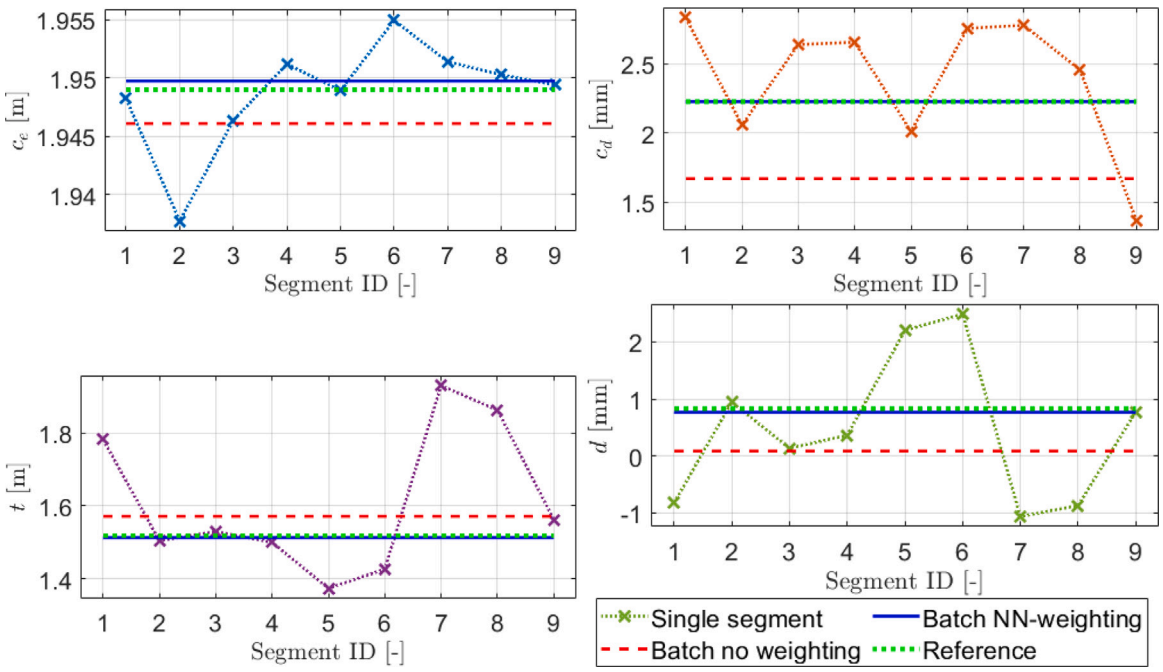


Fig. 17. Estimated parameters of the example batch.

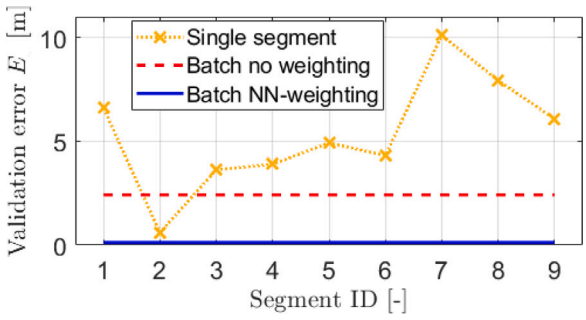


Fig. 18. Validation errors of the example batch.

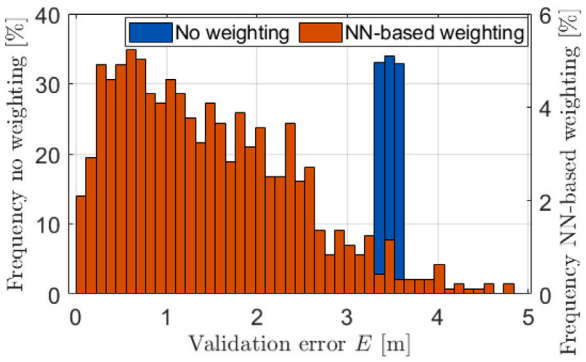


Fig. 19. Validation errors of the no weighting and our proposed NN-based weighting cases for the models in the $[3.3, 3.6]$ m group.

-5.47 mm, d : -0.062 mm s^2/m. The individual segment estimations with the single GN method have a mean error of 2.99 mm, 0.44 mm, 149.3624 mm, 1.07 mm s^2/m with a standard deviation of 4.76 mm, 0.49 mm, 199.24 mm, 1.29 mm s^2/m for the c_e, c_d, t, d vehicle parameters, respectively. This proves that the model can be appropriately calibrated even from such 9 segments whose individually optimal estimated parameters have a major error, which verifies the idea of relative weight integration.

6.5. Calibration results with the proposed algorithm

The previous section illustrates the calibration with the proposed architecture in detail while in this the robustness is examined, with the analysis of only the resulted validation error of all of the 20000 batches. Since the batches are quite various in terms of validation error (the errors are in a range of $[0.03, 5.92]$ m in the no weighting case), the batches are grouped according to this error to illustrate the performance of the method well. The batches fall into 20 bins with a width of 0.3 m. The "no weighting" case error on which the grouping is based is also referred to as the base error.

6.5.1. Calibration results of one error group

The validation errors of the models in the $[3.3, 3.6]$ m error group are presented in Fig. 19 with histogram. The model calibrations improve substantially by utilizing the estimated $\hat{w}_n$ weights by the neural network. The validation errors from the $[3.3, 3.6]$ m range in the no weighting case decrease to the mean of 1.47 m with a standard deviation of 0.98 m in the proposed NN-based weighting case. The 1.47 m as an absolute localization error of wheel odometry will be discussed in the next section. Regarding the calibration of the model, note that it is not certain for all of these batches (with errors between $[3.3, 3.6]$ m by raw calibration), that the unbiased parameter estimation resulting error close to 0 m is even achievable only with the modification of the w_n relative weights. Due to the supervised learning technique, it is more realistic to compare the value of improvement with the label weights.

For this reason, and to allow the opportunity for common comparison between groups with widely different errors, relative improvement metrics is also introduced as follows,

$$RI = \frac{E(\hat{\theta}_{No\ weighting}) - E(\hat{\theta}_{NN\ weighting})}{E(\hat{\theta}_{No\ weighting})} \cdot 100 \quad [\%], \tag{31}$$

where $E(\hat{\theta}_{No\ weighting})$ and $E(\hat{\theta}_{NN\ weighting})$ are the validation error of the no weighting and the proposed NN-based weighting cases, respectively.

This improvement for the $[3.3, 3.6]$ m error group can be found in Fig. 20 with histogram, and the values for the label weights are

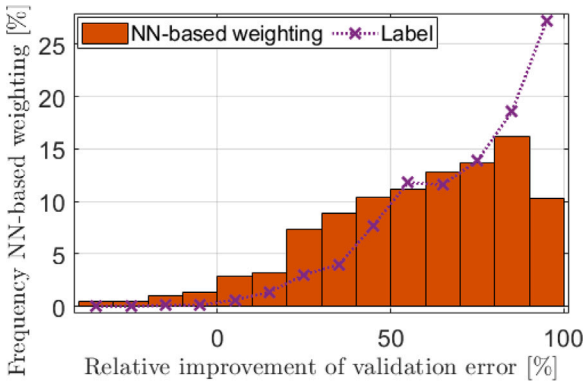


Fig. 20. Relative improvement (31) of our proposed NN-based weighting method for the models in the [3.3, 3.6] m error group.

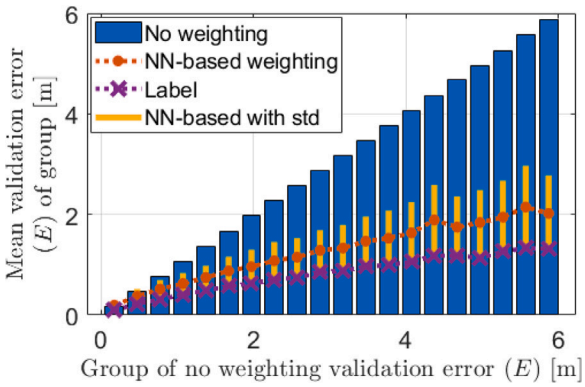


Fig. 21. Validation errors of the no weighting and our proposed NN-based weighting cases for all groups.

illustrated as well. The mean error improvement of 1.98 m by the proposed method is equivalent to an average relative improvement of 57.64% with a standard deviation of 28.30%, while the mean relative increase with the labels is 72.19% with a standard deviation of 22.08%. Thus, the estimation accuracy is doubled with the relative weights by the neural network for this group. However, there are batches resulting in worse calibration with the estimated weights. The bottom left corner of Fig. 20 shows that there is not any label inducing worse calibration. However, it has been mentioned in the analysis of Fig. 13 that the average weight estimation error for 3.71% of the batches is greater than 0.205, which are considered outliers. Consequently, the appearance of calibrations with larger errors is not surprising. The percentage of worse calibrations is 3.44% for this group which matches with the outlier ratio, therefore these calibration faults can be traced back to some inaccuracy in the weight estimation. Since trying countless other forms of neural networks has not improved the weight estimation, these few worse calibrations should be handled as a trade-off in the context of the proposed method.

Nevertheless, due to the applied supervised learning technique, the performance level of the labels can be handled as an achievable maximum. With the proposed NN-based weighting method, including the calibrations with higher error, 79.88% of this possible improvement is achieved, which means that the reduction in error from the [3.3, 3.6] m to the 1.47 m can be evaluated as a significant advancement.

6.5.2. Calibration results of all error groups

In this section, the calibration results with the validation error are presented for all error groups to test the robustness of the method. The base error spreads until 5.92 m, thus the 20000 batches are grouped into 20 error bins.

The validation errors can be found in Fig. 21, where each column represents a large number of batches (these could be shown in the same way as Group 14 in the previous section), thus the standard deviations are also illustrated beside the mean values. Generally, the proposed neural network-based method reduces the error by a huge amount, the average of the means is 1.27 m from 3 m, and *e.g.* at the worst group, the [5.45, 6.00] m error decreases to 2.02 m on average. The standard deviation increases almost linearly from 0.2 to 1.5 m with the base error of the groups. As it has been pointed out in the previous section, unbiased estimation is not certainly achievable for all batches, accordingly the error values spread between the base value of the group and zero, see Fig. 19. Taking this into account, the 1 m standard deviation cannot be considered outstanding.

Due to this spread and because the validation error is bounded from below, it is more interesting which upper error level contains a given percentage of the calibrated models. These levels in case of 4 percentage values for 7 error groups are shown in Table 3. The first row

Table 3
Upper validation error levels that contain a given percentage of the model calibrations with the proposed method.

		Group of no weighting error (E)						
		0.45	1.35	2.25	3.15	4.05	4.95	5.85
Per. of models	33%	0.21	0.47	0.67	0.75	0.85	1.04	0.92
	50%	0.30	0.67	0.97	1.16	1.33	1.64	1.67
	66%	0.40	0.88	1.28	1.59	1.92	2.15	2.56
	90%	0.70	1.33	2.00	2.61	3.25	3.66	3.93

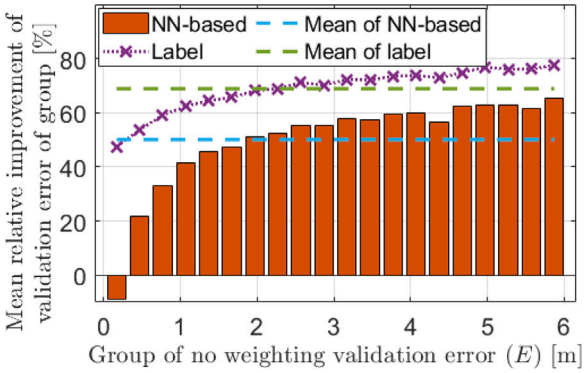


Fig. 22. Relative improvement (31) of our proposed NN-based weighting method for all groups.

illustrates that regardless of the base error (even at the worst [5.70, 6.00] m error group), 1/3 of the calibrated models with NN-based weights have below than 1 m validation error on the 300 m long segments. The values of the second row are equal to the groups' median. These are lower than the mean values of the groups, shown in Fig. 21, due to the piling up at the zero value. The difference increases from 0.1 to 0.3 m along the low and high base errors of the groups, respectively. Thus, the average calibration error can be considered to be less than the mentioned 1.27 m, using the median values of only 1.08 m. The last two rows demonstrate the robustness of the proposed method since the levels containing 66% of the calibrations are lower than half of the base error, and at 90% of the models, the error is below 4 m for the worst group as well, equivalent with 2/3 of the error in the no-weighting case.

The relative improvement (31) is a pleasing indicator of our algorithm, it can be found for the error groups in Fig. 22. The improvement compared to the base error is 50.1% with the estimated weights on an average but with the lack of consistency along the errors. Moreover, the variety is not random, the relative improvement increases with the validation error but in a damped way. For the groups with a

higher base error, *e.g.* more than 3 m, it verifies our assumption that a few bad segments of the batch could result in incorrect parameter estimation with a huge error, but if a correspondent algorithm is able to detect these and estimate the relative weights properly, the calibration performance is improved significantly.

However, for groups with low base error, *e.g.* less then 1 m, the proposed method is less effective, and the RI metrics even become negative for the group with $[0.00, 0.30]$ m error. Although this is an unpleasant outcome, the base errors of these 3 groups are fundamentally low with the mean of $0.15, 0.45, 0.75$ m, and these are changed to $0.19, 0.37, 0.52$ m with the estimated $\widehat{\omega}_n$ relative weights. These denote parameter estimation with high accuracy, thus the reduced improvement property of the proposed algorithm for low error groups is negligible and remains an acceptable trade-off.

Overall, the results demonstrate that the model calibration with the integrated NN-based relative weight estimation reaches low errors consistently or significantly reduces the error for the high base error groups.

7. Summary

7.1. Evaluation of the calibrated wheel odometry-based localization

The best way to evaluate odometry-based localization is to assume GNSS outage. The wheel odometry model has a 22.30 m validation error with the uncalibrated nominal setting, which makes it unusable for vehicle localization. With our proposed method, the error is only around 1.2 m on average on 300 m long segments, thus it is worth integrating into the state estimation layer of a vehicle. Only for evaluation, the calibrated models are tested on segments with different lengths as well. On the 300 m long calibration segments, the mean errors are 1.27 m and 2.48°. These are decreased to 0.64 m – 1.75° and 0.25 m–1.15° on 200 and 100 m long measurement sections, respectively (values are expressed relative to the error of the optimal model, as for the validation error).

The estimation performance with the trained model is also compared to other existing solutions from the related works. For a fair comparison, only works that similarly also apply offline training on a saved dataset before the online operations are considered. The comparison is performed with GNSS outages test on 100, 200, and 300 m long segments, respectively. In our method, the test segments are regenerated, in the others, linear scaling from the presented length is carried out.

Fig. 23 illustrates that the proposed method outperforms both methods in the 100 and 200 m case and performs similarly to the best of the others on 300 m. However, there are differences in the data used. In Onyekpe et al. (2021) only the speed of the 4 wheels are used for the localization, while in He et al. (2023) signals of an IMU are also included in the localization system, and in both works a single GNSS antenna is utilized in the training phase (beside the wheel and IMU signals). In the proposed method, the localization is based only on the two rear wheel signals, but acceleration values are also included. Furthermore, both methods operate with an end-to-end neural network for the prediction, while in the proposed method, parameter identification is performed with a neural network-based weighting, but the applied vehicle model is improved by the integration of lateral dynamics and vertical load transfer. Finally, the localization algorithm in the proposed method can be completed with an end-to-end prediction besides the model-based estimation that could increase accuracy.

Since wheel odometry is a relative pose estimation method, it is worth evaluating the errors as a speed sensor and comparing them to other ones. For example, the 0.25 m – 1.15° mean position and orientation errors on the 100 m long (7 s in integration time) segments, correspond to an angular velocity sensor with 0.003 rad/s, and a speed sensor with 0.035 m/s unknown bias. Due to the quadratic drift of the accelerometer, the 0.25 m error in 7 s is equivalent to 0.006 m/s^2 unknown bias, at 14.28 m/s average speed. These 0.035 m/s, 0.003 rad/s, and 0.006 m/s^2 uncertainties certainly exceed the accuracy of a mid-price IMU and GNSS sensor.

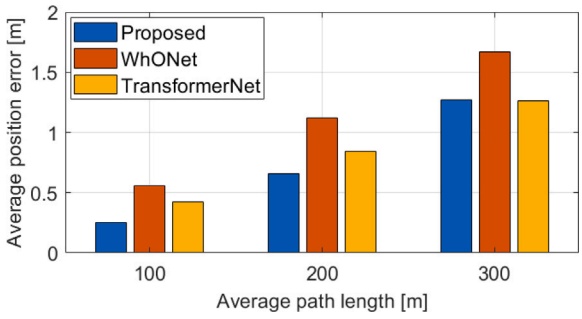


Fig. 23. Comparison of the localization accuracy of the proposed method in GNSS outages case with (Onyekpe et al., 2021) (WhoNet) and He et al. (2023) (TransformerNet).

7.2. Discussion and conclusions

In this paper, a novel method was presented for the calibration of the wheel odometry model of an autonomous vehicle. The idea is to integrate relative weights for the segments forming a batch in the Gauss–Newton parameter estimation algorithm to emphasize the impact of the good and mitigate the distortion of the bad measurements. While the base idea of weighting between the measurements is not new and simple enough, the realization that can be implemented online on vehicles raises challenging questions. Our solution is built around the idea that more powerful computers are available in modern self-driving vehicles or in the cloud, thus larger amounts of data and computationally intensive algorithms can be designed to improve the performance of control algorithms.

In the paper, it is illustrated with tests and examples that the relative weights have a crucial impact on the calibration accuracy. Thus, it is worth integrating into the online calibration. However, the construction of the algorithm that can predict the appropriate weights can be formulated only offline. Nevertheless, it is demonstrated that a machine learning-based approach is adequate, and its supplementary tasks, such as proper input feature determination and label generation, can be fulfilled with a combination of classic algorithms. Furthermore, the architecture can be expanded with an end-to-end method which can provide a complementary signal to predict the error terms that are unmodeled in the vehicle model.

The main contribution of this work is the proposed calibration architecture that integrates a neural network for estimating the relative weights. The paper describes in detail the two most critical parts of network development workflow, the input formulation and label generation, which are challenging tasks if the online operation with onboard signals is a required capability. The effectiveness of the algorithm is illustrated with experimental results from a real vehicle that demonstrates the significant improvement of the calibration accuracy using the same amount of online data. Thus, the calibrated wheel odometry model is an adequate choice to improve the state estimation of a self-driving vehicle, and in parallel, it is still a cost-effective method. Furthermore, the motion estimation with this accuracy can be used to estimate the noise of the GNSS or the IMU biases.

Moreover, no model-specific consideration or necessity of particular trajectories is applied, thus, the proposed method could be adapted to other parameter estimation tasks probably, which generates a bit of theoretical contribution to the topic of parameter identification.

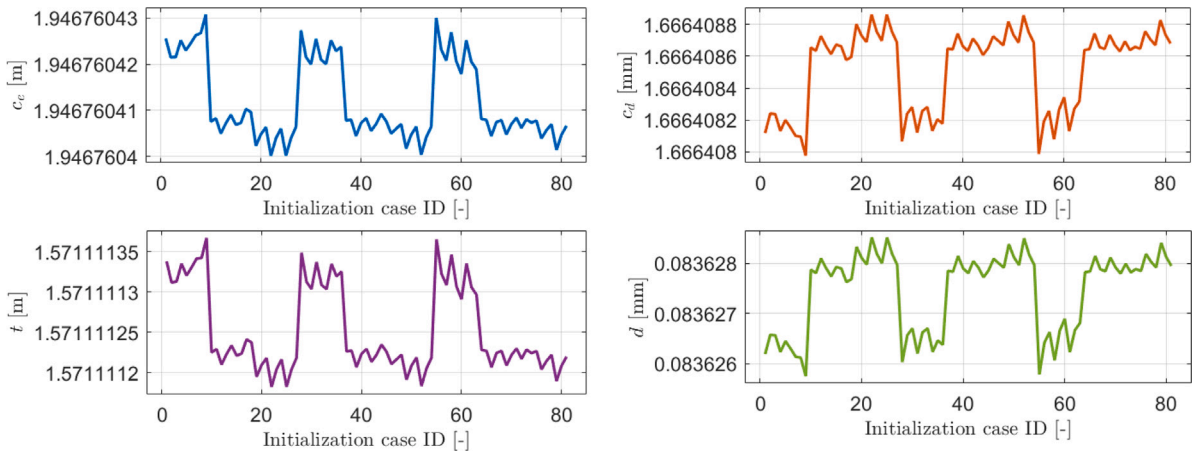


Fig. 24. Final estimated parameters of the test of the 81 initial parameter guess.

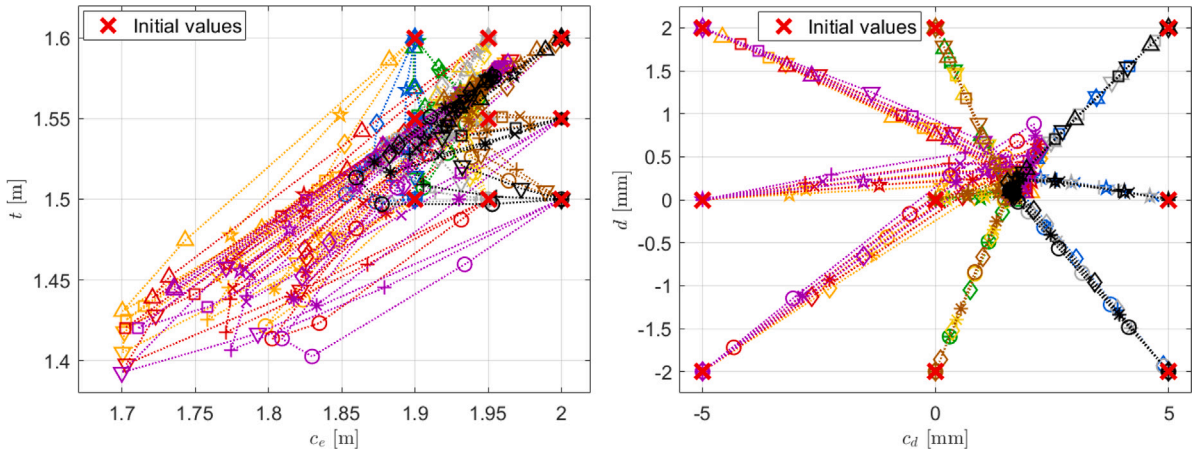


Fig. 25. Parameter convergence plots of the test of the 81 initial parameter guess.

7.3. Future works

Although the results presented are promising, there are still more opportunities for further development, which can be examined from the perspective of the learning method applied.

If the weighting knowledge from the offline recorded data is obtained via supervised learning, label generation is always a core improvement question since it is an upper limit for the calibration performance. In the case of the examined calibration task, the difficulty arises from the existence of several w locally optimal weight vectors of (21), *i.e.* substantially different w_n settings result in similar parameter estimation for the batch. The open question is how to determine the ones for the batches that can be represented with a common neural network.

The design of the neural network can be upgraded from the input side as well. Fig. 22 illustrates that utilizing the label weights, the improvement is 68% on average while the proposed method results in 50%. Regardless of how the labels are determined, the estimation performance can be further increased by mitigating this gap. An obvious solution is to increase the data fed into the net, *e.g.* by adding the individual optimal parameters of the segments, but be careful to avoid overfitting to the currently optimal vehicle parameters directly.

Another way to prevent the limitation of labels is not to use them at all. In this case, the reinforcement learning method should be applied, however, it requires a dynamic environment with states and actions whose formulation from a parameter identification task generates further questions.

CRediT authorship contribution statement

Máté Fazekas: Conceptualization, Investigation, Methodology, Software, Validation, Writing – original draft. **Péter Gáspár:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Appendix

Convergence and stability analysis of the GN nonlinear estimation method

Convergence plots of the Gauss–Newton method

See Fig. 27.

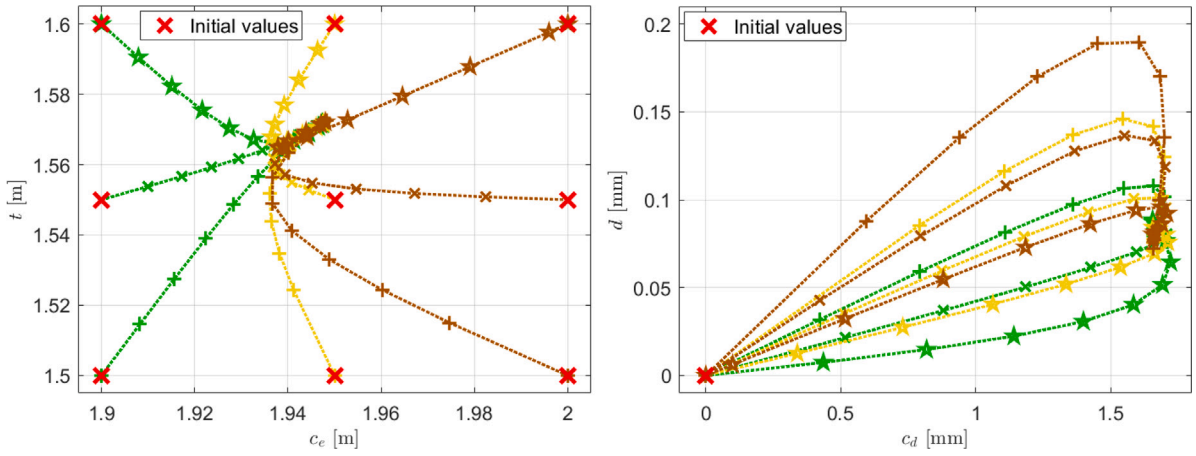


Fig. 26. Parameter convergence plots of the test of the various initial parameter guess when the c_d and d start from 0 as used in practice.

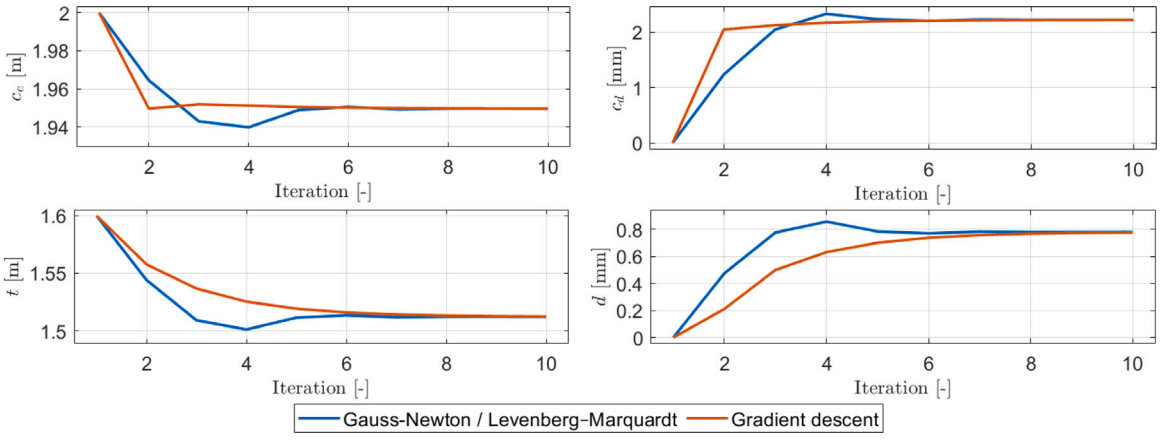


Fig. 27. Calibration with different nonlinear least squares method.

References

Antonelli, Gianluca, Chiaverini, Stefano, 2007. Linear estimation of the physical odometric parameters for differential-drive mobile robots. Auton. Robots 23, 59–68.

Antonelli, Gianluca, Chiaverini, Stefano, Fusco, Giuseppe, 2005. A calibration method for odometry of mobile robots based on the least-squares technique: Theory and experimental validation. IEEE Trans. Robot. 21 (5), 994–1004.

Bevly, David M., Ryu, Jihan, Gerdes, J. Christian, 2006. Integrating INS sensors with GPS measurements for continuous estimation of vehicle sideslip. IEEE Trans. Intell. Transp. Syst. 7 (4), 483–493.

Borenstein, Johann, Feng, Liqiang, 1996. Measurement and correction of systematic odometry errors in mobile robots. IEEE Trans. Robot. Autom. 12 (6), 869–880.

Brossard, Martin, Bonnabel, Silvere, 2019. Learning wheel odometry and IMU errors for localization. In: International Conference on Robotics and Automation. ICRA.

Brunker, Alexander, Wohlgemuth, Thomas, Frey, Michael, Gauterin, Frank, 2017. GNSS-shortages-resistant and self-adaptive rear axle kinematic parameter estimator (SA-RAKPE). In: 28th IEEE Intelligent Vehicles Symposium.

Brunker, Alexander, Wohlgemuth, Thomas, Frey, Michael, Gauterin, Frank, 2018. Odometry 2.0: A slip-adaptive UIF-based four-wheel-odometry model for parking. IEEE Trans. Intell. Transp. Syst. 4 (1), 114–126.

Caltabiano, Daniele, Muscato, Giovanni, Russo, Francesco, 2004. Localization and self-calibration of a robot for volcano exploration. In: IEEE International Conference on Robotics and Automation. pp. 586–591.

Caron, Francois, Duflos, Emmanuel, Pomorski, Denis, Vanheeghe, Philippe, 2006. GPS/IMU data fusion using multisensor Kalman-filtering: Introduction of contextual aspects. Inf. Fusion 7 (2), 221–230.

Censi, Andrea, Franchi, Antonio, Marchionni, Luca, Oriolo, Giuseppe, 2013. Simultaneous calibration of odometry and sensor parameters for mobile robots. IEEE Trans. Robot. 29 (2), 475–492.

CleanTechnica, 2019. Tesla's new HW3 self-driving computer. https://cleantechnica.com/2019/06/15/teslas-new-hw3-self-driving-computer-its-a-beast-cleantechnica-deep-dive/.

Falco, Gianluca, Pini, Marco, Marucco, Gianluca, 2017. Loose and tight GNSS/INS integrations: Comparison of performance assessed in real urban scenarios. Sensors 17 (2).

Fariña, Bibiana, Acosta, Daniel, Toledo, Jonay, Acosta, Leopoldo, 2023. Improving odometric model performance based on LSTM networks. Sensors 23 (2).

Fazekas, Máté, Gáspár, Péter, Németh, Balázs, 2021a. Calibration and improvement of an odometry model with dynamic wheel and lateral dynamics integration. Sensors 21 (2).

Fazekas, Máté, Gáspár, Péter, Németh, Balázs, 2021b. Odometry model calibration for self-driving vehicles with noise correction. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS.

Fazekas, Máté, Gáspár, Péter, Németh, Balázs, 2022. Wheel odometry model calibration with input compensation by optimal control. In: 10th IFAC Symposium on Advances in Automotive Control. AAC.

Fazekas, Máté, Németh, Balázs, Gáspár, Péter, Sename, Olivier, 2020. Vehicle odometry model identification considering dynamic load transfers. In: 28th Mediterranean Conference on Control and Automation. MED, pp. 19–24.

Funk, Niklas, Alatur, Nikhilesh, Deuber, Robin, 2017. Autonomous electric race car design. In: International Electric Vehicle Symposium.

Gao, Zhouzheng, Ge, Maorong, Li, You, Chen, Qijin, Zhang, Quan, Niu, Xiaoji, Zhang, Hongping, Shen, Wenbin, Schuh, Harald, 2018. Odometer, low-cost inertial sensors, and four-GNSS data to enhance PPP and attitude determination. GPS Solut. 22 (57), 147–159.

Goldberg, David E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Gutiérrez, Joaquín, Apostolopoulos, Dimitrios, Gordillo, José Luis, 2007. Numerical comparison of steering geometries for robotic vehicles by modeling positioning error. Auton. Robots 23, 147–159.

He, Ke, Ding, Haitao, Xu, Nan, Guo, Konghui, 2023. Wheel odometry with deep learning-based error prediction model for vehicle localization. Appl. Sci. 13 (9).

Jung, Daun, Seong, Jihoon, bae Moon, Chang, Jin, Jiyong, Chung, Woojin, 2016. Accurate calibration of systematic errors for car-like mobile robots using experimental orientation errors. Int. J. Precis. Eng. Manuf. 17 (9), 1113–1119.

Kelly, Alonzo, 2004. Linearized error propagation in odometry. Int. J. Robot. Res. 23 (2), 179–218.
Kingma, Diederik P., Ba, Jimmy Lei, 2015. ADAM: A method for stochastic optimization. In: 3rd International Conference for Learning Representations.
Kochem, M., Neddenriep, R., Isermann, R., Wagner, N., 2002. Accurate local vehicle dead-reckoning for a parking assistance system. In: American Control Conference. pp. 4297–4302.
Kulathunga, Nalinda, Ranasinghe, Nishath Rajiv, Vrinceanu, Daniel, Kinsman, Zackary, Huang, Lei, Wang, Yunjiao, 2021. Effects of nonlinearity and network architecture on the performance of supervised neural networks. Algorithms 14 (2).
Kümmerle, Rainer, Grisetti, Giorgio, Burgard, Wolfram, Simultaneous parameter calibration, localization, and mapping. Adv. Robot. 26 (17), 2021–2041.
Larsen, Thomas Dall, Bak, Martin, Andersen, Nils A., Ravn, Ole, 1998. Location estimation for an autonomously guided vehicle using an augmented Kalman filter to autocalibrate the odometry. In: International Conference on Multisource-Multisensor Information Fusion.
Lee, Kooktae, Chung, Woojin, Yoo, Kwanghyun, 2010. Kinematic parameter calibration of a car-like mobile robot to improve odometry accuracy. Mechatronics 20 (5), 582–595.
Lemmer, László, Heb, R., Krauss, Markus, Schilling, Klaus, 2010. Calibration of a car-like mobile robot with a high-precision positioning system. In: 2nd IFAC Symposium on Telematics Applications.
Ljung, Lennart, 2010a. Approaches to identification of nonlinear systems. In: 29th Chinese Control Conference.
Ljung, Lennart, 2010b. Perspectives on system identification. Annu. Rev. Control 34 (1), 1–12.
Martinelli, Agostino, 2002. The odometry error of a mobile robot with a synchronous drive system. IEEE Trans. Robot. Autom. 18 (3), 399–405.
Martinelli, Agostino, Siegwart, Roland, 2006. Observability properties and optimal trajectories for on-line odometry self-calibration. In: IEEE Conference on Decision and Control. pp. 3065–3070.
Martinelli, Agostino, Tomatis, Nicola, Siegwart, Roland, 2007. Simultaneous localization and odometry self calibration for mobile robot. Auton. Robots 22, 75–85.
Mathworks, 2021. MATLAB global optimization toolbox: Genetic algorithm.
Mathworks, 2022. MATLAB deep learning toolbox.
Maye, Jérôme, Sommer, Hannes, Agamennoni, Gabriel, Siegwart, Roland, Furgale, Paul, 2016. Online self-calibration for robotic systems. Int. J. Robot. Res. 35 (4), 357–380.
Mohamed, Sherif A.S., Haghbayan, Mohhamad-Hashem, Westerlund, Tomi, Heikkonen, Jukka, Tenhunen, Hannu, Plosila, Juha, 2019. A survey on odometry for autonomous navigation systems. IEEE Access 7, 97466–97486.
Navone, Alessandro, Martini, Mauro, Angarano, Simone, Chiaberge, Marcello, 2023. Online learning of wheel odometry correction for mobile robots with attention-based neural network.
Onyekpe, Uche, Palade, Vasile, Herath, Anuradha, Kanarachos, Stratis, Fitzpatrick, Michael E., 2021. WhONet: Wheel odometry neural network for vehicular localisation in GNSS-deprived environments. Eng. Appl. Artif. Intell. 105, 104421.
Roy, Nicholas, Thrun, Sebastian, 1999. Online self-calibration for mobile robots. In: IEEE International Conference on Robotics and Automation. pp. 2292–2297.
Rudolph, Alexander, 2003. Quantification and estimation of differential odometry errors in mobile robotics with redundant sensor information. Int. J. Robot. Res. 22 (2), 117–128.
Scaramuzza, Davide, Fraundorfer, Friedrich, 2011. Visual odometry - Part I: The first 30 years and fundamentals. IEEE Robot. Autom. Mag. 18 (4), 80–92.
Schoukens, Johan, Ljung, Lennart, 2019. Nonlinear system identification: A user-oriented roadmap. IEEE Control Syst. Mag. 39 (6), 28–99.
Seegmiller, Neal, Rogers-Marcovitz, Forrest, Miller, Greg, Kelly, Alonzo, 2013. Vehicle model identification by integrated prediction error minimization. Int. J. Robot. Res. 32 (8).
Tangirala, Arun K., 2015. Principles of System Identification. CRC.
Thrun, Sebastian, et al., 2006. Stanley: The robot that won the DARPA Grand Challenge. J. Field Robotics 23 (9).
Toledo, Jonay, Piñeiro, Jose D., Arnay, Rafael, Acosta, Daniel, Acosta, Leopoldo, 2018. Improving odometric accuracy for an autonomous electric cart. Sensors 18 (1), 200–2015.
Xu, Haoming, Collins, John James, 2009. Estimating the odometry error of a mobile robot by neural networks. In: International Conference on Machine Learning and Applications.
Yu, Hao, Wilamowski, Bogdan M., 2018. Levenberg-Marquardt training. In: Intelligent Systems. CRC Press, pp. 12:1–16.
Zhang, Zhihuang, Zhao, Jintao, Huang, Changyao, Li, Liang, 2021. Learning end-to-end inertial-wheel odometry for vehicle ego-motion estimation. In: 5th CAA International Conference on Vehicular Control and Intelligence.