

Article

Improving Odometric Model Performance Based on LSTM Networks

Bibiana Fariña * , Daniel Acosta, Jonay Toledo and Leopoldo Acosta 

Computer Science and System Department, Universidad de La Laguna, 38200 San Cristobal de La Laguna, Spain
* Correspondence: bfarinaj@ull.edu.es

Abstract: This paper presents a localization system for an autonomous wheelchair that includes several sensors, such as odometers, LIDARs, and an IMU. It focuses on improving the odometric localization accuracy using an LSTM neural network. Improved odometry will improve the result of the localization algorithm, obtaining a more accurate pose. The localization system is composed by a neural network designed to estimate the current pose using the odometric encoder information as input. The training is carried out by analyzing multiple random paths and defining the velodyne sensor data as training ground truth. During wheelchair navigation, the localization system retrains the network in real time to adjust any change or systematic error that occurs with respect to the initial conditions. Furthermore, another network manages to avoid certain random errors by using the relationship between the power consumed by the motors and the actual wheel speeds. The experimental results show several examples that demonstrate the ability to self-correct against variations over time, and to detect non-systematic errors in different situations using this relation. The final robot localization is improved with the designed odometric model compared to the classic robot localization based on sensor fusion using a static covariance.

Keywords: mobile robot; self-localization; odometry; sensor fusion; long short-term memory



Citation: Fariña, B.; Acosta, D.; Toledo, J.; Acosta, L. Improving Odometric Model Performance Based on LSTM Networks. *Sensors* **2023**, *23*, 961. <https://doi.org/10.3390/s23020961>

Academic Editor: Natividad Duro Carralero

Received: 22 December 2022

Revised: 8 January 2023

Accepted: 9 January 2023

Published: 14 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Auto-localization is an essential component of autonomous robots. It enables the robot to navigate safely and accurately by providing real-time information about its pose and surroundings. To achieve this, auto-localization systems typically use fusion algorithms to combine data from multiple sensors, such as cameras, LIDARs, and GPS, providing a robust and accurate localization module even in dynamic environments. The fact of receiving several data based on different operating principles, allows the system to become resilient to sensor failure, ensuring that the robot can continue to navigate effectively.

There are many fusion techniques used in auto-localization, including the Kalman filter. This last one requires each sensor to provide a noise covariance, which describes its measurement accuracy and determines its influence on the final localization. For non-linear systems, there are variants of the Kalman filter, such as the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). The EKF linearizes the system using the Taylor series, but it can be less effective in highly non-linear systems [1]. The UKF, on the other hand, uses the unscented transform to fit the probability density distribution of non-linear equations, which allows it to avoid the loss of higher-order terms that can occur with linearization [2]. This makes the UKF a simple, fast, and precise option for non-linear systems [3–5].

In order to achieve accurate state estimation for auto-localization systems, it is not enough to simply use the right fusion techniques. The reliability of the sensor data is also crucial. Sensors can be easily affected by noise, and overcoming this uncertainty is essential for improving the efficiency of the auto-localization.

One of the key sensors used in wheeled mobile robots is the odometric sensor, which is a precise, cheap, and easy-to-process sensor. However, the pose calculated by this sensor

is incremental, which can lead to the accumulation of small deviations over time. This can result in significant errors in the pose estimation. Additionally, the odometric sensor is sensitive to random errors, such as drifting, which can make it difficult to track and potentially cause inaccurate poses.

Considering these limitations and how artificial intelligence is one of the fastest-growing technologies in mobile robotics, this paper presents the application of neural networks to odometry, improving the data accuracy and also reducing the sensor errors. In this way, the encoder data are processed by a neural network to include effects that a static model does not represent.

This method is implemented in an autonomous wheelchair that has been designed to transport people with severe disabilities [6]. The wheelchair is controlled by the user indicating the destination, while the navigation module will automatically execute the optimal path, using the sensors for the reconstruction of the environment and the obstacles detection [7]. In this way, to achieve safe autonomous navigation, the localization is an essential part of the chair movement, since its failure would affect all other subsystems.

The wheelchair has the following set of sensors:

- **Odometry:** it estimates the wheel robot movement by optical encoders coupled to the motor. In the prototype, the resolution of the encoder is 8800 pulses per revolution with a resolution of 0.04 degrees. The pose is calculated by the wheel movement knowing the radii and distance between them.
- **Light Detection and Ranging (LIDAR):** they are two Sick TiM 551 sensors, with a maximum distance of 10 m, an angular resolution of 1 degree and a viewing angle of 270 degrees. The relative movement of the robot can be calculated using two consecutive laser scans. The measurement accuracy is estimated by the ICP (Iterative Closest Point) algorithm [8,9]. It will return a low covariance value if two consecutive sweeps have many singular points that ensure their precise matching. However, if they have few singular points, the covariance will be high.
- **Inertial Measurement Unit (IMU):** an MPU9250 sensor which consists of an accelerometer, a magnetometer and a gyroscope. Our tests indicate that the magnetometer and the accelerometer are too imprecise to consider its measurements so only the gyroscope will be used. Likewise, the sensor covariance is estimated by setting a static value.

The robot state $\tilde{x}_k(V, W)$ is estimated by a UKF. In this case, the algorithm estimates the state at the next instant (k) by using the uniform rectilinear motion model and considering the sensor data $z_k(V, W)$ and their covariances $(COV_k^{v/w})_z$ as measurements. The output of the filter not only includes the estimated robot state but also its covariance $(COV_k^{v/w})_{filter}$, which measures the precision of the filter at that moment. Moreover, the pose (X, Y, θ) is calculated by integrating the output state of the filter. Figure 1 shows the wheelchair and a schema of the implemented localization where our technique is applied.

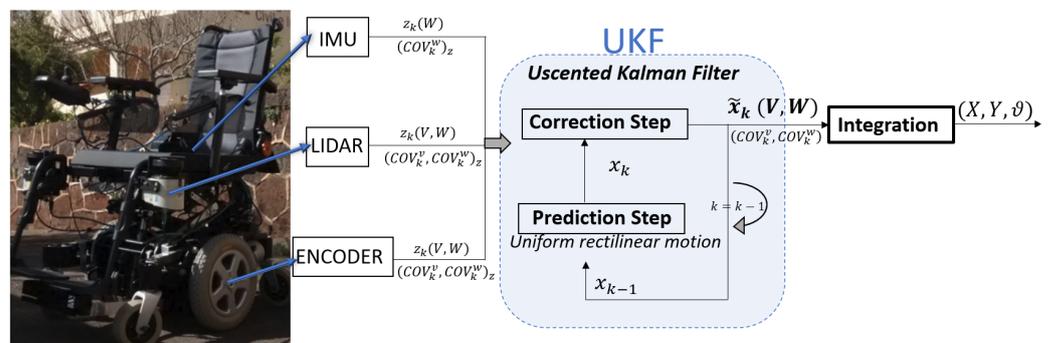


Figure 1. The localization system proposed for the wheelchair.

Accordingly, our paper focuses on the following key points: the robot self-localization problem applied to autonomous wheelchair and the LSTM (long short-term memory) neural network technique to improve the odometric sensor. The paper is structured as follows. Section 2 includes a review of the studies that have been completed prior to this work. Section 3 describes the wheelchair kinematics model. Likewise, Section 4 explains the methodology carried out to improve the accuracy of the odometric sensor. The experiments using the designed model are shown in Section 5, and the conclusions are summarized in Section 6.

2. Previous Work

The study of assistive robots is gaining attention for the possible social, economic, and scientific applications. There are investigations that collect the design of different types of autonomous wheelchair. Ref. [10] designs an autonomous wheelchair capable of segmenting passable areas and anomalies on the roads through deep learning. Refs. [11,12] show the characteristics of two wheelchairs for the transport of people in predefined hospital. Ref. [13] describes an adaptive neural control system for governing the movements of a robotic wheelchair. Likewise, [14,15] use an ROS (Robotic Operative System) in the wheelchair for the architecture of their modules.

Regarding novel techniques in the current literature on self-localization, there have been significant advances in the application of artificial intelligence (AI) to mobile robotics. This has led to the development of more intelligent and autonomous robots that can navigate complex environments and complete tasks more efficiently and accurately. Ref. [16] describes a survey of the different matching learning techniques for motion planning and control for mobile robots. In [17,18], a track fusion algorithm based on the LSTM method are proposed achieving better results in the fusion effect. Likewise, Refs. [19,20] present a deep reinforcement learning approach for the motion planning of autonomous robots in dynamic surroundings.

Machine learning is also applied in the odometry pose estimation to improve the results of using the traditional method by geometric equations. In [21], a feedforward neural network model is used in order to learn the odometric model from data. However, these sensors contain errors that need to be corrected in order to obtain a reliable pose. Real-time calibration of the structure's parameters is important to adapt to any changes in the model over time. There are various techniques in the literature that attempt to adjust these parameters. Refs. [22,23] solves the car odometric systematic errors by using the final pose difference in a predefined path. Other works try to correct them in real time by the design of an AKF (Augmented Kalman Filter) [24] or by estimating the deviation of the radius nominal value by using a marginalized particle filter (MPF) [25].

With respect to non-systematic errors, such as slip, there are some methods that attempt to reduce them. Ref. [7] uses a Doppler speed sensor in order to measurement the odometric noise depending on the data quality and obtain a dynamic odometric covariance. Likewise, Refs. [26,27] try to correct them by reducing their influence in the output filter. They design a real-time outlier detection in the observations, applying a saturation function to the residual. Moreover, Refs. [28,29] try to avoid them by using the dynamic structure model. This last one proposes an slip ratio estimator based on the motion equations, the input torque and encoders data.

Taking into account the AI advances in the field of self-localization and the previously mentioned limitations of the odometric sensor, this paper presents a new approach based on LSTM networks for predicting odometry poses. The proposed system is able to self-calibrate in real-time through network retraining and another LSTM network is used to learn and compensate for non-systematic errors in the final localization.

3. Robot with Differential Kinematics

The wheelchair has differential kinematics. It consist of two drive wheels on a common axis. The robot can move and turn thanks to the independent wheel driving forwards

or backwards. Figure 2 shows the scheme of a differential mobile robot with radii $R_{r/l}$ and a distance D between them. The robot movement between two time instants would correspond to a circle trajectory with radius ρ around the point ICC (Instantaneous Centre of Curvature). From the wheel speeds $V_{r/l}$, it is possible to estimate the linear (V) and angular (W) robot velocities, as well as its pose (X, Y) and its orientation (θ).

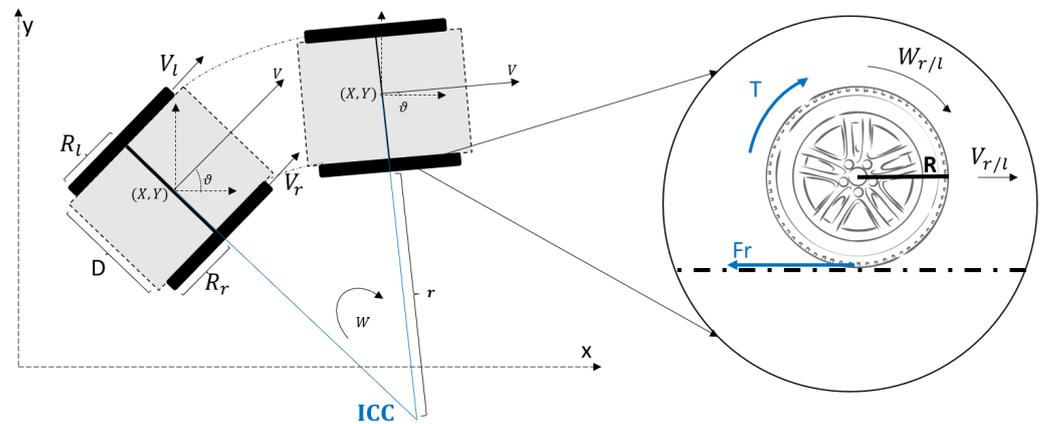


Figure 2. Differential kinematics schema for a mobile robot.

Figure 2 shows the robot wheel dynamics, where T corresponds to the motor torque that generates a linear ($V_{r/l}$) and angular ($W_{r/l}$) velocities. A friction force (F_r) opposes to the movement, it can have a significant impact on the behavior and movement of the robot. These forces depend on the environment in which the robot operates, and small variations in their values can affect the robot's movement. For example, certain types of robots, such as wheeled mobile robots, are very sensitive to small changes in the speed of each wheel. Even minor errors in the relative speeds between the wheels can affect the robot's path. These robots are also sensitive to small variations in the ground plane, and they may need additional wheels (swivel casters) for support.

3.1. Wheel-Encoder-Based Traditional Odometry

The wheelchair's pose at each instant can be obtained from the wheel speeds following the scheme of the Figure 2 by using odometric equations. In this case, we can estimate the respective wheel speeds by using the counts provided from the encoder sensor as follow,

$$V_{ok}^l = \frac{2\pi R_l \Delta ct_k^l}{encRes \Delta T}; \quad V_{ok}^r = \frac{2\pi R_r count_k^r}{encRes \Delta T} \quad (1)$$

where $V_{ok}^{l/r}$ corresponds to the left and right wheel velocities. $\Delta ct_k^{l/r}$ are the number of each encoder ticks that the electronics receive in time period (ΔT), and $encRes$ is the encoder resolution in one turn. The wheels parameters are the corresponding radii R_r and R_l and the distance between the wheels D .

Considering the circle trajectory with radius ρ around the ICC point, we can estimate the linear (V_k) and angular (W_k) robot velocities (Equation (2)). This calculation is obtained by geometric equations knowing that the linear velocity of a circumference multiplied by the radius is equal to the angular velocity.

$$\frac{V_{ok}^r}{r} = W_k; \quad \frac{V_{ok}^l}{\rho + D} = W_k; \quad \frac{V_k}{\rho + D/2} = W_k \quad (2)$$

$$W_k = \frac{(V_{ok}^l - V_{ok}^r)}{D}; \quad V_k = \frac{(V_{ok}^l + V_{ok}^r)}{2}$$

Once the current robot velocities $(V, W)_k$ has been estimated, it is necessary to integrate, considering the time interval ΔT , to obtain the estimated robot pose $(X, Y, \theta)_{k+1}$ at each moment,

$$\begin{aligned} \theta_{k+1} &= \theta_k + W_k \Delta T \\ X_{k+1} &= X_k + V_k \cos(\theta_k) \Delta T \\ Y_{k+1} &= Y_k + V_k \sin(\theta_k) \Delta T \end{aligned} \tag{3}$$

3.2. Wheel-Motion Pattern Effect

The behavior of wheeled mobile robots depends on the environment characteristics. The contact type between the wheel and the ground and its friction force value significantly influence the robot’s movement.

In ideal conditions, regardless of the terrain type, the robot speed depends on the motor torque and its power. An increase in the motor power will lead to an increase in angular wheel speed that will be translated into linear velocity. The opposite case is given to reduce the robot speed. However, there are situations where, depending on the current robot speed and the adhesion degree with the terrain, the robot can slip or skid, producing two situations. The first is due to the motor’s dead zone that does not produce enough torque to move the wheel, causing the wheel to stall instead of turning with the robot in motion. The second occurs when the wheel rotates freely, slipping and without any robot movement.

From this assumption, we anticipate that there is a logical correlation between the angular wheel speed and the motor power consumed. This relation disappears when non-systematic errors occur due to the type contact with the ground.

Figure 3 shows graphically an example of the relationship between the motor power and its angular velocity, given by the encoder sensor counts. In Figure 3a, there is no significant slip, following wheel speed and motor power, a similar pattern, increasing and decreasing in a correlated way. However, in Figure 3b there is a slip, highlighted by the green lines, where, although the power increases, the real angular velocity decreases, contrarily to the expected effect. Using this property, the motor power can be used to detect non-systematic errors in the odometric system. There is not a clear mathematical relation between motor power and speed in order to detect non systematic errors, so a Neural Network is used to learn this relation from data.

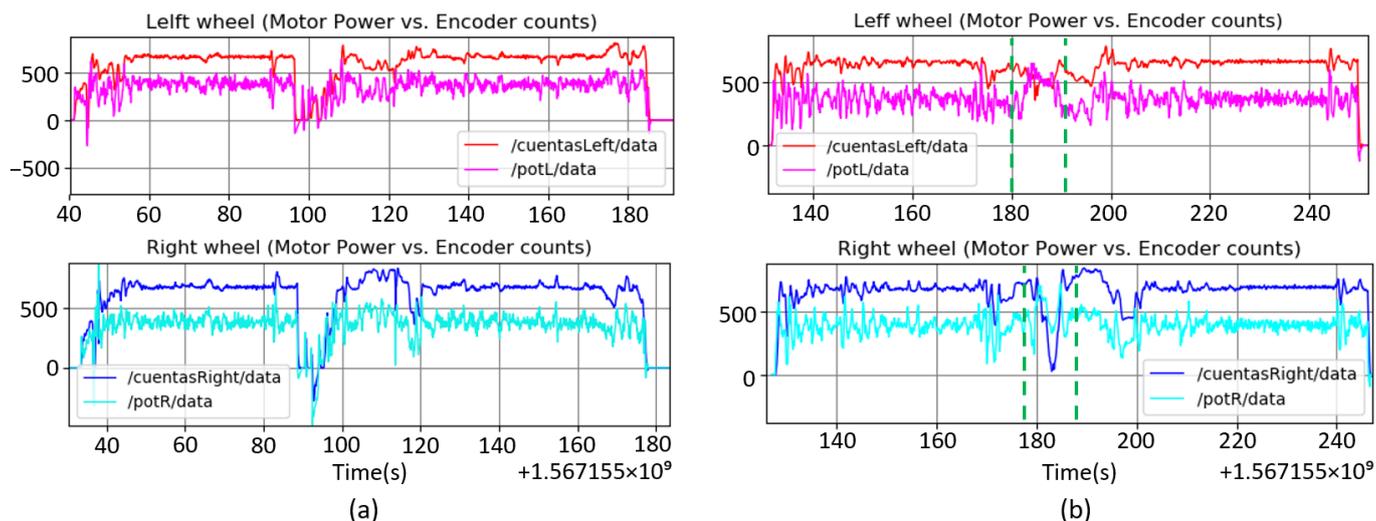


Figure 3. Motor power vs. encoder counts for both wheels.(a) Without slipping and (b) with slipping.

4. Odometry System Proposed

In this localization system, the data accuracy provided by the odometric sensor is improved using neural networks that learn from the robot behavior. It avoids using manually estimated parameters, improving and correcting some errors, which in the mathematical model are not appreciable and, therefore, difficult to adjust.

The proposed model consists of two neural networks that have, as input, the incremental encoder counts ($\Delta ct_k^{r/l}$) of the right and left wheels, and provides, as output, the robot state, that are the linear (V_k) and angular (W_k) speeds for the respective network. We decided to use two neural networks for each output instead of a single one in order to allow each network to specialize in predicting a specific output and converge more quickly and easily. This is because in the following sections we will have to retrain them online and we need the network to quickly adapt and converge to any changes that may occur in the model. Using a single network with two outputs makes retraining much more difficult and it does not adequately adjust to our desired requirements. This has also been tested experimentally, and we found that the architecture proposed resulted in more precise execution.

The training has been carried out offline by creating a dataset from the information of multiples paths with different characteristics. During the training, the speed data provided by a velodyne HDL 32 sensor at each moment have been collected as the desired output of the robot state. It has been installed in the upper part of the chair and consists of 32 LIDARs with a precision of 2 cm that enables the calculation of the state with high reliability using the LOAM slam algorithm [30,31].

We decided to use recurrent networks, specifically, the LSTM-type [32] was chosen over other models, such as MLP, because of its ability to learn and remember patterns over time. This is particularly useful as it can take into account long-term patterns in the encoder data sequence. Moreover, during the neural network design, we compared the use of other models using the wheelchair dataset and it demonstrates a better effectiveness during training and validation. For the LSTM network, it has been considered a time window of 20 samples that are provided by the sensor at a frequency of 10 Hz. Figure 4 shows the structure carried out to obtain the linear (V_k) and angular (W_k) velocity at each time k . In order to train the designed model, pre-processing data are necessary. First, the raw data were resized into a tensor whose dimension was (samples, time step, features), which in our case was (10,115,20,3). The inputs at one moment correspond to the encoder count increments and the speed at the previous time ($\Delta ct^r, \Delta ct^l, V/W$) $_{k-1}$. The outputs are the lineal V_k and angular W_k speeds of the robot. Moreover, the input must be normalized to the same scale by eliminating the mean and scaling to a standard deviation equal to 1. So, the samples of an input type are pre-processed using:

$$z = \frac{s-\mu}{\sigma} \quad \text{with} \quad \mu = \frac{1}{N} \sum_{i=1}^N s_i \quad \sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (s_i - \mu)^2} \quad (4)$$

where z is the new normalized value and s is the raw sample. μ is the mean and σ is the standard deviation of the input, and N correspond to the samples dimension.

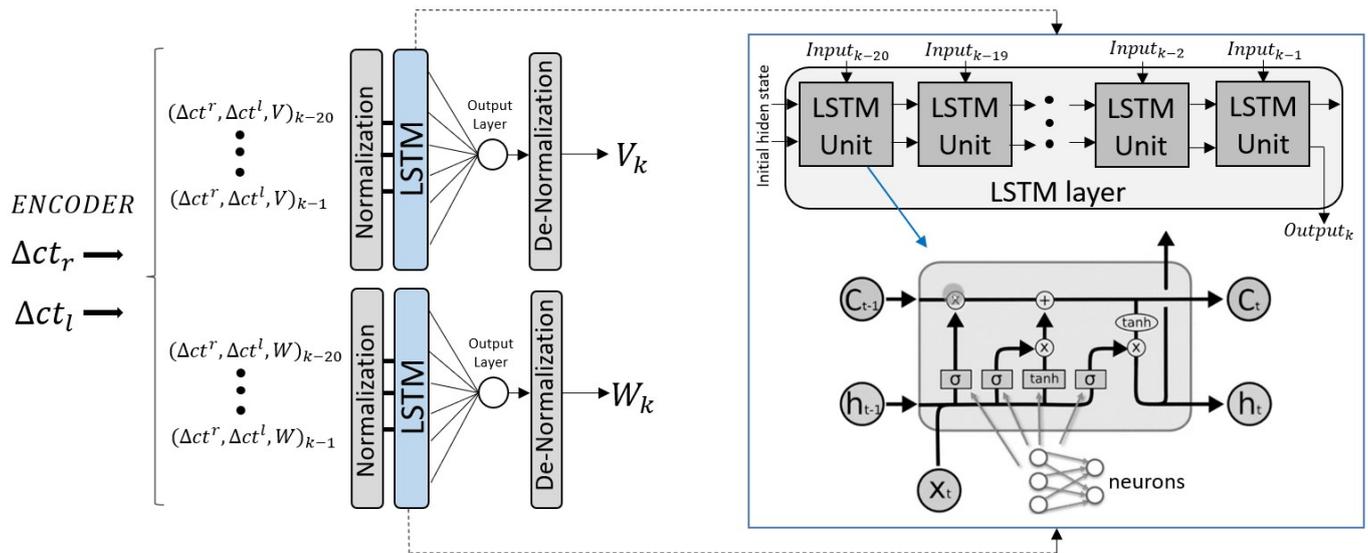


Figure 4. Schema of the odometric model proposed using LSTM networks.

Regarding the implemented LSTM layer, it analyzes an input data stream and produces a prediction. In this structure, data are introduced sequentially in an LSTM unit (cell) that can be divided into different parts. In each cell, x_t corresponds to the input at the current time, h_{t-1} is the previous cell output and C_{t-1} is the cell state. First, the forget gate controls how much information about C_{t-1} is relevant from the inputs h_{t-1} and x_t , by using a sigmoid layer (σ),

$$f_t = \sigma(\omega_f[h_{t-1}, x_t] + b_f) \quad (5)$$

where ω_f represents the weight matrix and b_f the bias. f_t value in each dimension is in the range (0, 1). The information will be forgotten when f_t is close to 0, and the information will be retained when is close to 1.

Subsequently, the gateway is responsible for processing the current input and updating the relevant information. It has two parts, the first (i_t) controls how much of the input is stored using a sigmoid layer and the second (\tilde{C}_t) generates a new current state candidate by using the tanh function,

$$i_t = \sigma(\omega_i[h_{t-1}, x_t] + b_i) \quad \tilde{C}_t = \tanh(\omega_c[h_{t-1}, x_t] + b_c) \quad (6)$$

where ω_i and ω_c represent weight matrix and b_i and b_c represent the bias. From this information, the cell state can be updated from the previous one (C_{t-1}) and from the new generated candidate (\tilde{C}_t),

$$\tilde{C}_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (7)$$

Finally, the output gate generates the cell output (h_t) from the selection of the relevant information in the current cell state,

$$h_t = \sigma(\omega_o[h_{t-1}, x_t] + b_o) * \tanh(C_t) \quad (8)$$

where ω_o is the weight matrix and b_o its bias.

Therefore, considering the Figure 4, the data are processed and normalized, then divided into a 70% of training and a 30% as test data. In this case, the LSTM is 1 layer with 5 neurons in each cell, chosen ad hoc, providing the best training and validation results for different tests. This state is passed to a full connected output layer with 1 neuron for a single output and linear activation. Regarding the training, the ADAM optimizer algorithm and the MAE (mean absolute error) metric has been used with a learning rate of 0.001, a batch size of 10, 150, and 200 epochs depending on the model. Figure 5 shows the loss in

both training and validation, using the MAE errors obtained with respect to the epochs for each model, V and W.

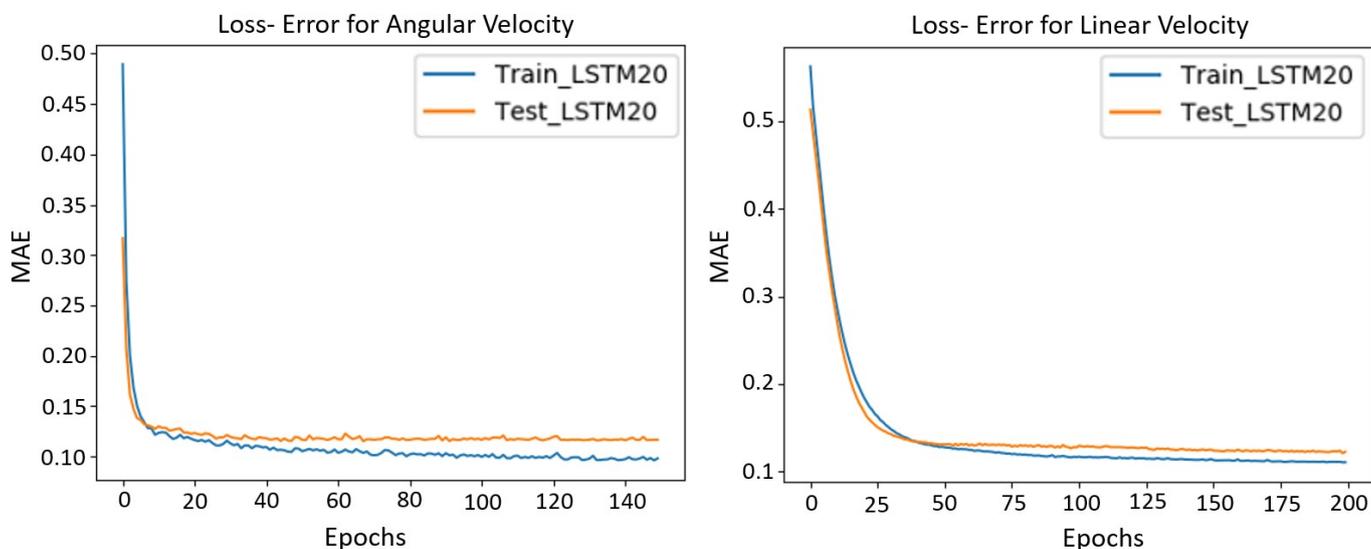


Figure 5. Loss function per epochs using the MAE for the angular and linear velocities.

4.1. Odometry Error Covariance Modeling

The odometric system can be improved using an LSTM network to detect non-systematic errors that occur in unpredictable environments. For example, local bumps, holes, rocks, and non-level terrain can cause non-systematic errors. Using gyro sensors to detect these errors has shown good performance, as demonstrated in previous research [33,34]. However, there are other cases that are more difficult to detect, such as wheel slippage on level terrain. In these cases, a non-systematic detection method is necessary.

As explained in the first section, the fusion algorithm estimates the robot localization from different sensors, where each measurement is defined with a covariance that characterizes its confidence. In the case of odometry, it takes the speed covariances $(COV_k^V, COV_k^W)_{odom}$. Then, the filter prioritizes the measurements with a small covariance and avoids those with greater values. So, it is important to dynamically adjust the covariance of the odometric sensor to assign a high number in the presence of non-systematic errors and, thus, reduce its influence on the final localization result, compared to other sources. This variable covariance is designed based on the assumptions in Section 3.2, where we demonstrate how the wheel speeds depend on the motor torque and its power, and how the adhesion degree with the terrain causes the wheel to slip, affecting this dependence. Therefore, we need a system that is able to learn the relationship between the power of each wheel and its corresponding angular velocity in order to detect when this relationship is not fulfilled and is in an error state. In this case, the system will act accordingly by assigning a very large static value to all elements in the covariance matrix, considering that they are stochastically independent random variables.

Figure 6 shows the method implemented on the wheelchair for this purpose. It consists of an LSTM neural network that takes as input the powers of each wheel (P^r, P^l), the encoder increment counts ($\Delta ct^r, \Delta ct^l$), and the sensor covariance in the previous measurements $(COV_{k-n}^V, COV_{k-n}^W)$. As output, it is able to estimate the next step covariance (COV_k^V, COV_k^W) . The data have been preprocessed using the same method discussed earlier, organized into time windows with memory of 20 samples, and normalized to have a standard deviation of one and a mean equal to zero. In our case, several samples have been collected with different slip types and whose input dimension is (12442,20,5), which have been divided into 70% for training and 30% for validation. The training has been carried out offline by using the velodyne sensor data. The network structure is a LSTM layer consisting of 10 neurons and a full connected output layer with 1 neuron for a single

output and ReLU activation. We considered the ADAM optimizer algorithm and the MAE metric with a learning rate of 0.001, 120 epochs, and a batch size of 10. A de-normalization process is required in the final phase to obtain the estimated covariance data.

Figure 6 also shows the variance of the angular odometry velocity estimated by our designed network with respect to time, where a slippage happens (non-systematic error) in the period 400–405 s. As we can see, the covariance remains at a low value when the sensor operates correctly and increases considerably when it is in an error state. This information allows the UKF filter to weight the input data in real time. If the LSTM Network detects a failure condition, the covariance will increase, so the data coming from odometry will be considered by the filter with a small weight. The odometry data together with this variable covariance will be used in a subsequent filter to be fused with other sensors and its influence on the final localization result will depend on that estimated covariance.

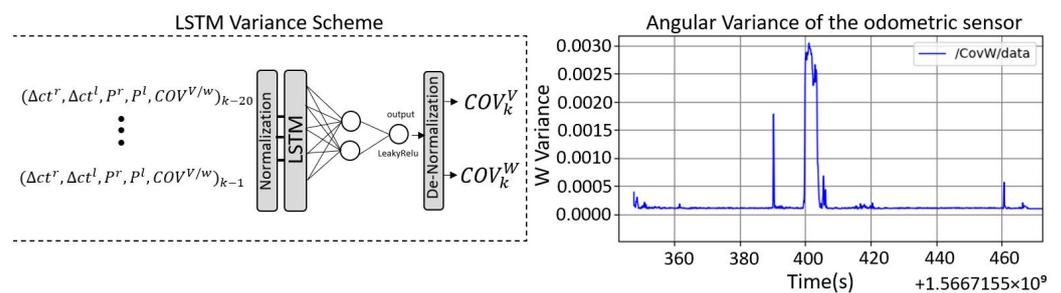


Figure 6. Scheme implemented to avoid non-systematic odometry errors, and the evolution of the angular velocity variance when a slippage happens in an example path.

4.2. Real Time Auto-Calibration

Real-time calibration of the model is crucial when designing an odometry system. It must be able to detect and correct any systematic error that may arise in relation to the offline-trained model. For example, a sudden change in the model may occur due to a shift in the weight of passenger, which can significantly impact the model accuracy. Over time, other situations, such as a change in tire pressure, can also affect the model performance, causing errors in the estimated localization. In order to ensure high accuracy, the system should be able to adapt to these changes in real-time by fine-tuning its model.

The scheme used for this calibration is shown in Figure 7. In this case, the neural network models for both angular and linear velocity estimation are retrained in real-time using input data from encoders $(\Delta ct^r, \Delta ct^l)$ and the output from the UKF filter speeds $(V, W)_k$. It is an iterative process where the data are stored in real-time, and when 300 samples are collected, the network weights are adjusted accordingly. These samples are only stored whenever the covariance of the filter (COV_{Filter}) and the variable covariance of the odometry (COV_{Odom}) are small, which ensures that the data used for re-training are reliable and accurate. In this way, the model is changed only when the robot is well localized, avoiding introducing errors in the retraining process.

Figure 7 also shows the error in angular velocity over time with respect to the ground truth (velodyne) when there is a change in the model, causing the error to increase. These graphs compare the behavior of the odometric system with and without re-training. Without considering the calibration, the model does not detect any change and the error grows and remains constant. In case of the real-time calibration, the system is able to detect these changes and adjust the model by retraining, reducing the relative error with respect to the velodyne sensor.

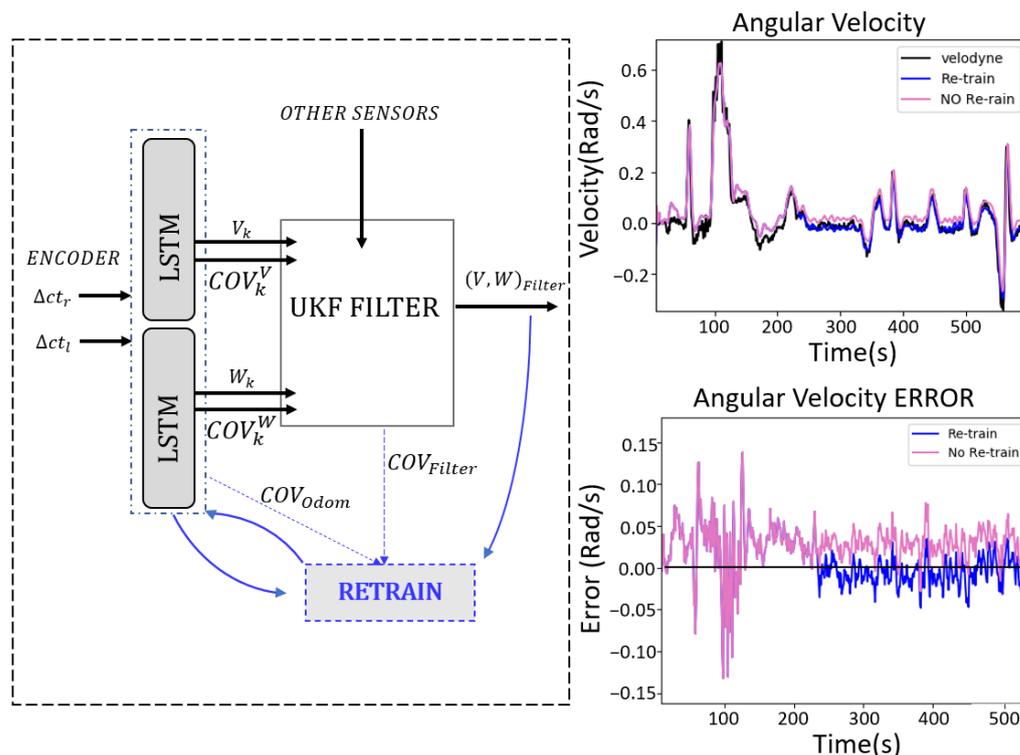


Figure 7. Retraining scheme for odometric model calibration. The error using the model with real-time retraining is smaller than using the model without it.

5. Results

The validation of the implemented wheelchair localization system was demonstrated through the study and analysis of several paths with different characteristics. These experiments were carried out in the corridor of the robotics laboratory at the University of La Laguna. In addition, to evaluate the proposed odometric model, different parameters and speeds were used on the paths to demonstrate the functionality of its variable covariance and the online calibration, as well as to ensure that a wide range of circumstances were covered. The effectiveness of the calibration was tested by varying the passengers and altering the diameters of the wheels through inflation. Furthermore, in the experiments to detect errors from the environment, the wheelchair speed has been increased in various sections and we have used other wheels with greater wear to generate slipping. We have also created these situations by applying pressure on one side, preventing the rotation of a wheel while allowing the other to turn freely.

On the one hand, Figure 8a,b shows two different example paths where the estimated pose of the odometry using LSTM networks (green line) and the pose obtained using traditional odometry (blue line) have been compared with the reference data from the velodyne (black line). As can be observed, the use of our model significantly improves the estimated pose accuracy, providing results that are closer to the ground truth.

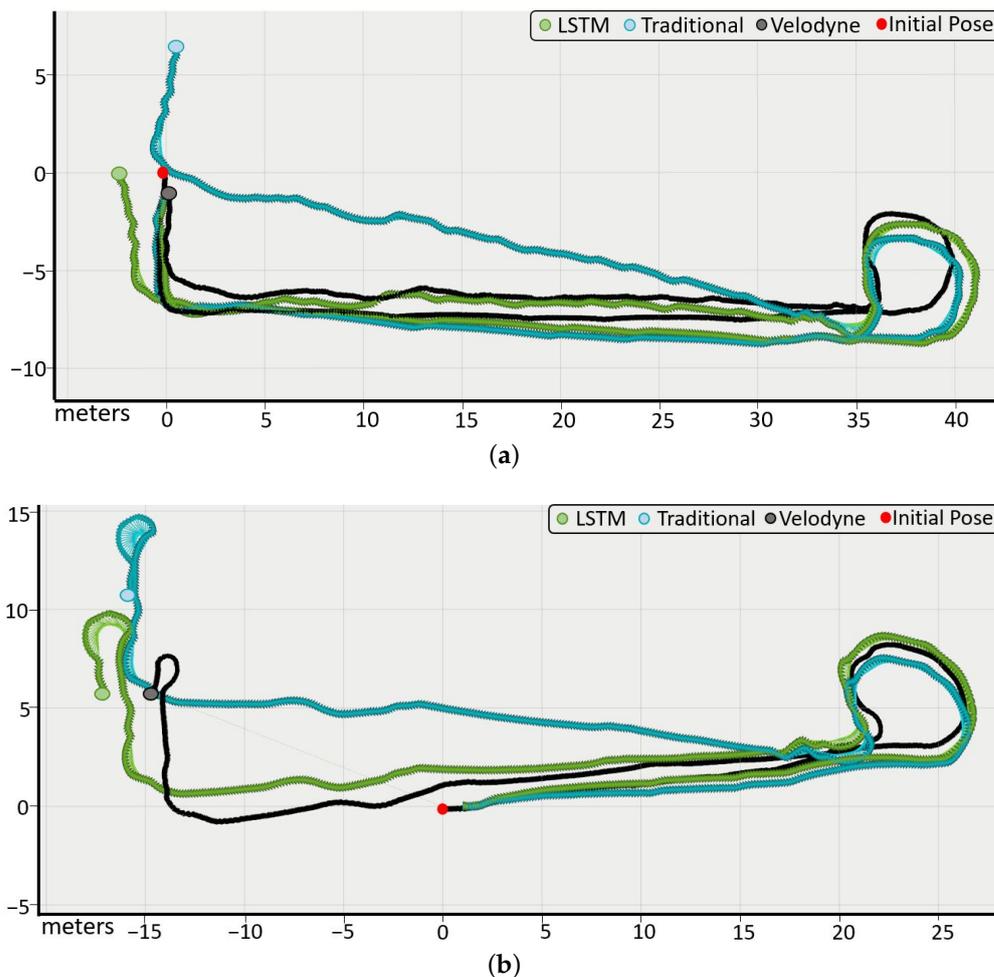


Figure 8. Paths to validate the odometry model proposed. Odometry with LSTM networks (green), traditional odometry (blue), and ground truth (black).

This fact is also demonstrated numerically in Table 1. In this case, multitude of trajectories have been executed and the odometry error with respect to the velodyne has been collected for both cases, using our LSTM network and using traditional model. We have analyzed the errors at each time in the velocities (V, W) and in the robot pose (X, Y, θ). The table shows the mean squared error (RMSE), the mean absolute error (MAE), and the R squared, which are defined by the following formulae:

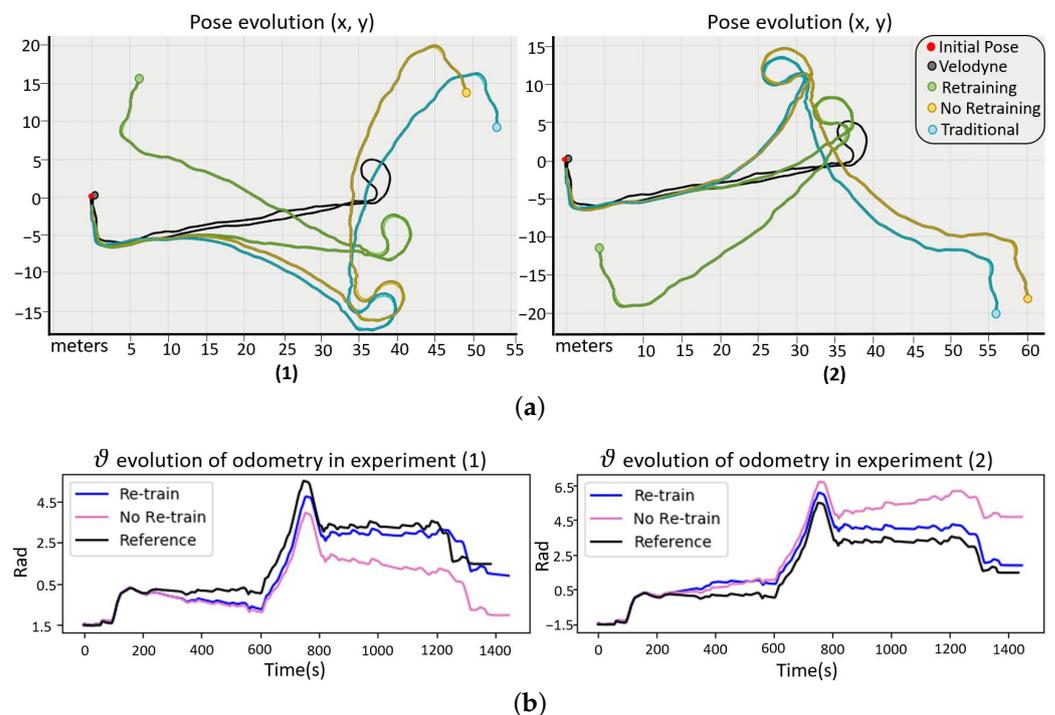
$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad MAE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (9)$$

where \hat{y}_i is the predicted value, y_i is the true value, n is the number of samples, and \bar{y} is the mean value of the true values. Based on the obtained values, the errors of our system are smaller than the traditional model. Therefore, despite the small difference between the errors, it demonstrates the effectiveness of our model in improving the odometry localization of the wheelchair.

Table 1. Error values obtained by comparing the LSTM odometry model and the traditional odometry with respect to the velodyne data.

		V (m/s)	W (rad/s)	X,Y (m)	Pose θ (rad)
LSTM odometry	MAE	0.04646	0.08103	1.56692	0.07131
	RMSE	0.10282	0.1791	2.076499	0.0864
	R_{score}	0.799122	0.84129	0.9569	-
	Accuracy %	79.91%	84.13%	95.69%	-
Global Model Accuracy (%)					86.57%
Traditional odometry	MAE	0.062744	0.102723	1.83725	0.1720
	RMSE	0.1237	0.200329	2.48464	0.2151
	R_{score}	0.6734	0.7953	0.9389	-
	Accuracy %	67.43%	79.53%	93.89%	-
Global Model Accuracy (%)					80.2%

On the other hand, the proposed real-time calibration model has been studied in several paths and with different changes in the system that have occurred both initially and during it. Figure 9 shows two examples where the characteristics of one wheel have been intentionally modified from the initial model to represent the odometry behavior when a systematic error arises. It causes the pose to deviate, generating an error in the velocities that, when integrated, accumulates over time in the pose. We compared the behavior of three different odometry models: the neural network-based odometry with real-time retraining (green line), odometry with neural networks but without retraining (yellow line), and traditional odometry (blue line). As reference, we consider the velodyne (black line).

**Figure 9.** Example paths to validate the calibration model by retraining in real-time. (a) Shows the position “X, Y” and (b) shows the orientation “ θ ”.

This figure shows how, thanks to retraining, the model is adjusted to new changes and corrects the pose increments of the trajectory, approaching the velodyne. Table 2 collects the errors obtained from several experiments comparing the proposed odometry model with and without retraining. Based on these results, the neural network-based odometric model and traditional odometry are unable to perceive these changes, increasing the error

indefinitely. If we consider the calibration, the error in the velocities is reduced. However, it does not directly correct the error pose, but rather gradually improves the pose increments at each time interval. Moreover, we are interested in correcting the velocities, which will be the input measurement data for the filter in estimating the final pose of the robot.

Table 2. Error values obtained by comparing the LSTM-odometry model with and without retraining considering the velodyne data as reference.

		V (m/s)	W (rad/s)	X,Y (m)	Pose θ (rad)
LSTM Retraining	MAE	0.0384577	0.06484	4.27604	0.56772
	RMSE	0.069028	0.115653	8.34219	0.629522
	R_{score}	0.848956	0.71317	0.25404	-
	Accuracy %	84.9%	71.32%	25.4%	-
Global Model Accuracy (%)					60.54%
LSTM No retraining	MAE	0.05496	0.07919	5.156277	1.5677
	RMSE	0.081812	0.22197	8.90407	1.7094
	R_{score}	0.67216	0.57198	0.03743	-
	Accuracy %	67.22%	57.2%	3.74%	-
Global Model Accuracy (%)					42.7%

Regarding the process implemented to reduce the influence of non-systematic errors, the output of the UKF filter has been studied in various experiments with different slip types. Figure 10 shows a path example where a considerable non-systematic error occurs. The green line represents the pose estimated by the odometry, the blue one the LIDAR, the red is the UKF localization, and the black line defines the ground truth. This error type causes the odometry pose to deviate significantly, and since it is a random error, it is impossible to correct its pose. The proposed solution is to modify the odometry covariance. During the corridor, the LIDAR sensor has a high covariance due to the low number of possible matching points and its estimated pose could be erroneous, however the odometry provides accurate results. In turn, due to the slip, the error in the odometric sensor grows and the LIDAR is more accurate.

Figure 10a shows the UKF-filter output when the traditional odometry is used with a static covariance manually adjusted. The result is worse since it either relies too much on odometry, underestimating the LIDAR or vice versa. Figure 10b shows the localization result of the UKF filter with the proposed LSTM neural network with non-systematic error detection and dynamic covariance estimation. During the slip, the power of the motors and wheel speed is not correlated, the error classifier detects a non-systematic error during the turn, and increases the covariance, reducing the odometry influence on the final localization during that error state. The final pose of the filter is close to ground truth, and clearly better than without the detector.

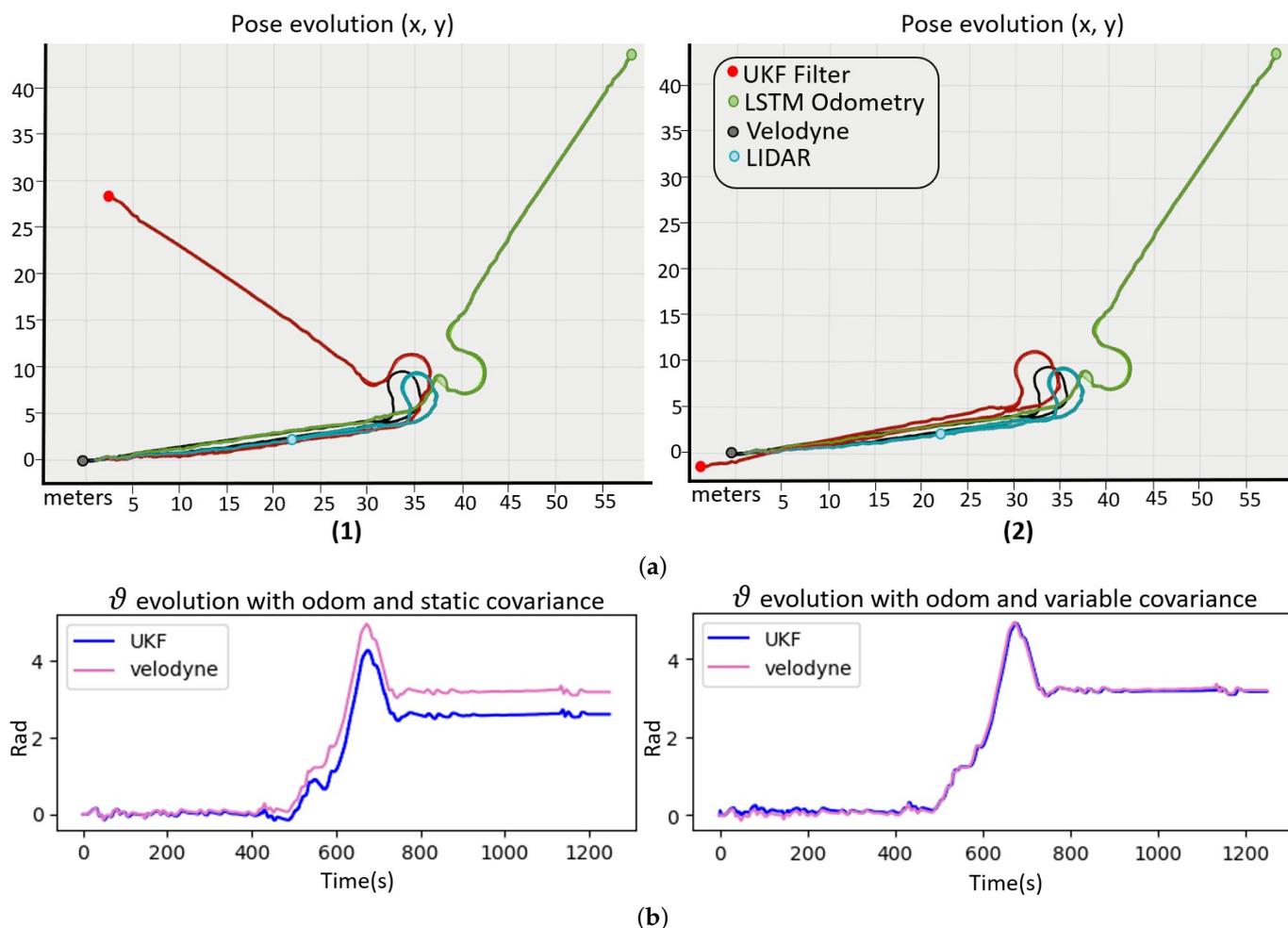


Figure 10. Localization system for a path where the red line is the filter output, the green is the odometry, the blue the LIDAR, and the black line for the ground truth. (a) Shows the position “X, Y” and (b) shows the orientation “ θ ”. (1) Odometry with static covariance and (2) odometry with variable covariance.

In addition to the graphical study, the errors of the UKF filter, with respect to the velodyne, have been collected in different experiments for both cases, using traditional odometry with static covariance and the proposed odometry with variable covariance. Table 3 shows the metric values used for its comparison that are MAE, RMSE, R squared, and we have also considered the normalized estimation error squared (NEES) using the filter results and the estimated odometry poses against the ground truth. This metric takes into account the state covariance and is defined as:

$$NEES = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^T P_i^{-1} (x_i - \hat{x}_i) \quad (10)$$

where n is the number of samples, x_i is the true state, \hat{x}_i is the estimated state, and P_i is the covariance matrix for the i th sample.

Therefore, the proposed odometry model with a variable covariance has been shown to improve the accuracy of the UKF filter in the presence of non-systematic errors. This has been demonstrated through graphical analysis and by comparing the errors of the UKF filter using the proposed odometry model and the traditional odometry model with static covariance. The NEES metric has also been used to evaluate the accuracy of the filter, with a lower NEES value indicating better performance.

Table 3. Error values obtained by comparing the UKF filter output with respect to the velodyne data in two cases. Using the odometry model with variable covariance and with static covariance.

		V (m/s)	W (rad/s)	X,Y (m)	Pose θ (rad)
LSTM odometry	MAE	0.04125	0.03149	1.4021	0.0547
	RMSE	0.03804	0.05464	2.8699	0.07037
	R_{score}	0.83329	0.80617	0.9279	-
	$NEES_{Odom}$	1087	1982	-	-
	$NEES_{Filter}$	1614	2963	-	-
	Accuracy %	83.33%	80.62%	92.8%	-
Global Model Accuracy (%)					85.58%
Traditional odometry	MAE	0.05333	0.0452	2.0592	0.38356
	RMSE	0.0661	0.071619	5.3323	0.4759
	R_{score}	0.7697	0.67634	0.678483	-
	$NEES_{Odom}$	6439	7650	-	-
	$NEES_{Filter}$	5986	4779	-	-
	Accuracy %	76.97%	67.63%	67.85%	-
Global Model Accuracy (%)					70.82%

6. Conclusions

This work presents a new odometry system implemented in an autonomous wheelchair. It consists of LSTM neural networks that are able to estimate the robot speed using the data from the encoder sensor. The real-time retraining allows the system to self-calibrate and adapt to changes in the defined model, further improving its performance. Likewise, it is able to reduce the influence of some non-systematic errors by training LSTM networks that learn the relationship between the power of the wheels and their angular velocity to design a variable covariance.

In conclusion, this system significantly improves the accuracy of the wheelchair estimated pose by the UKF filter, compared to the use of classic odometry methods. This is demonstrated through graphical analysis and numerical comparisons of the errors between the different methods. The results show how our system errors are smaller and closer to the velodyne data, providing more accurate final robot localization, particularly in presence of systematic and non-systematic errors. Overall, the proposed wheelchair localization provides a more robust solution for state estimation in challenging environments.

Author Contributions: Conceptualization, B.F., D.A., J.T. and L.A.; methodology, B.F., and D.A.; software, B.F., D.A., J.T. and L.A.; validation, B.F. and J.T.; formal analysis, D.A. and L.A.; investigation, B.F., D.A., J.T. and L.A.; data curation, D.A. and J.T.; writing—original draft preparation, B.F. and D.A.; writing—review and editing, B.F., J.T. and L.A.; visualization, J.T. and L.A.; supervision, L.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors gratefully acknowledge the contribution from the Spanish Ministry of Science and Technology under the SIRTAPE project DPI2017-90002-R. Author B.F. is partially supported by the grant FPU (Formación de Personal Universitario) of Spanish Ministry of Education.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chadaporn, K.; Baber, J.; Bakhtyar, M. Simple Example of Applying Extended Kalman Filter. In Proceedings of the 1st International Electrical Engineering Congress, Chiangmai, Thailand, 13–15 March 2013.
2. Wan, E.A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), Lake Louise, AB, Canada, 4 October 2000; pp. 153–158. [[CrossRef](#)]

3. Liu, H.; Hu, F.; Su, J.; Wei, X.; Qin, R. Comparisons on Kalman-Filter-Based Dynamic State Estimation Algorithms of Power Systems. *IEEE Access* **2020**, *8*, 51035–51043. [[CrossRef](#)]
4. Konatowski, S.; Kaniewski, P.; Matuszewski, J. Comparison of Estimation Accuracy of EKF, UKF and PF Filters. *Annu. Navig.* **2016**, *23*. [[CrossRef](#)]
5. Fariña, B.; Toledo, J.; Acosta, L. Sensor fusion algorithm selection for an autonomous wheelchair based on EKF/UKF comparison. In Proceedings of the 2022 10th International Conference on Control, Mechatronics and Automation (ICCMA), Luxembourg, 9–12 November 2022.
6. Arnay, R.; Hernández-Aceituno, J.; Toledo, J.; Acosta, L. Laser and Optical Flow Fusion for a Non-Intrusive Obstacle Detection System on an Intelligent Wheelchair. *IEEE Sens. J.* **2018**, *18*, 3799–3805. [[CrossRef](#)]
7. Fariña, B.; Toledo, J.; Estevez, J.; Acosta, L. Improving Robot Localization Using Doppler-Based Variable Sensor Covariance Calculation. *Sensors* **2020**, *20*, 2287. [[CrossRef](#)] [[PubMed](#)]
8. Bonnabel, S.; Barczyk, M.; Goulette, F. On the covariance of ICP-based scan-matching techniques. In Proceedings of the 2016 American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 5498–5503.
9. Brossard, M.; Bonnabel, S.; Barrau, A. A New Approach to 3D ICP Covariance Estimation. *IEEE Robot. Autom. Lett.* **2020**, *5*, 744–751. [[CrossRef](#)]
10. Wang, H.; Sun, Y.; Liu, M. Self-Supervised Drivable Area and Road Anomaly Segmentation Using RGB-D Data For Robotic Wheelchairs. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4386–4393. [[CrossRef](#)]
11. Alkhatib, R.; Swaidan, A.; Marzouk, J.; Sabbah, M.; Berjaoui, S.; O.Diab, M. Smart Autonomous Wheelchair. In Proceedings of the 2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART), Paris, France, 24–26 April 2019; pp. 1–5. [[CrossRef](#)]
12. Sinyukov, D.; Desmond, R.; Dickerman, M.; Fleming, J.; Schaufeld, J.; Padir, T. Multi-modal control framework for a semi-autonomous wheelchair using modular sensor designs. *Intell. Serv. Robot.* **2014**, *7*, 145–155. [[CrossRef](#)]
13. Boquete, L.; Barea, R.; García, R.; Mazo, M.; Sotelo, M.A. Control of a Robotic Wheelchair Using Recurrent Networks. *Auton. Robot.* **2005**, *18*, 1573–7527. [[CrossRef](#)]
14. Baltazar, A.; Petry, M.R.; Silva, M.F.; Moreira, A.P. Driverless Wheelchair for Patient’s On-Demand Transportation in Hospital Environment. Ph.D. Thesis, University Oporto, Porto, Portugal, 2020. [[CrossRef](#)]
15. Grewal, H.; Matthews, A.; Tea, R.; George, K. LIDAR-based autonomous wheelchair. In Proceedings of the 2017 IEEE Sensors Applications Symposium (SAS), Glassboro, NJ, USA, 13–15 March 2017; pp. 1–6. [[CrossRef](#)]
16. Xiao, X.; Liu, B.; Warnell, G.; Stone, P. Motion Planning and Control for Mobile Robot Navigation Using Machine Learning: A Survey. *arXiv* **2020**, arXiv:2011.13112. [[CrossRef](#)]
17. Zhang, C.; Hu, L.; Xu, Y.; Rasol, J.; Zhou, Q.; Zhang, Z. Multiple Sensor Track Fusion Algorithm Based on LSTM Network. In Proceedings of the 2022 International Conference on Image Processing and Media Computing (ICIPMC), Xi’an, China, 27–29 May 2022; pp. 154–158. [[CrossRef](#)]
18. Wang, M.; Xu, C.; Zhou, C.; Gong, Y.; Baochun, Q. Study on Underwater Target Tracking Technology Based on an LSTM–Kalman Filtering Method. *Appl. Sci.* **2022**, *12*, 5233. [[CrossRef](#)]
19. Everett, M.; Chen, Y.F.; How, J.P. Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1805.01956.
20. Kahn, G.; Abbeel, P.; Levine, S. *BADGR: An Autonomous Self-Supervised Learning-Based Navigation System*; IEEE: Piscataway, NJ, USA, 2020. [[CrossRef](#)]
21. Toledo, J.; Piñeiro, J.D.; Arnay, R.; Acosta, D.; Acosta, L. Improving Odometric Accuracy for an Autonomous Electric Cart. *Sensors* **2018**, *18*, 200. [[CrossRef](#)] [[PubMed](#)]
22. Lee, K.; Chung, W. Calibration of kinematic parameters of a Car-Like Mobile Robot to improve odometry accuracy. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 2546–2551. [[CrossRef](#)]
23. Borenstein, J. Experimental results from internal odometry error correction with the OmniMate mobile robot. *IEEE Trans. Robot. Autom.* **1998**, *14*, 963–969. [[CrossRef](#)]
24. Hassanzadeh, I.; Fallah, M. Design of Augmented Extended and Unscented Kalman Filters for Differential-Drive Mobile Robots. *J. Appl. Sci.* **2008**, *8*. [[CrossRef](#)]
25. Lundquist, C.; Karlsson, R.; Ozkan, E.; Gustafsson, F. Tire Radii Estimation Using a Marginalized Particle Filter. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 663–672. [[CrossRef](#)]
26. Haile, M.; Wang, Y. *Robust Extended Kalman Filtering for Systems With Measurement Outliers*; IEEE: Piscataway, NJ, USA, 2021. [[CrossRef](#)]
27. Ting, J.A.; Theodorou, E.; Schaal, S. A Kalman filter for robust outlier detection. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 1514–1519. [[CrossRef](#)]
28. Kim, J.; Kim, B.K. Cornering Trajectory Planning Avoiding Slip for Differential-Wheeled Mobile Robots. *IEEE Trans. Ind. Electron.* **2020**, *67*, 6698–6708. [[CrossRef](#)]
29. Kono, U.; Fujimoto, H.; Hori, Y. Localization of Wheeled Mobile Robots from Slip Ratio Estimation with Simple Model. In Proceedings of the 2021 IEEE International Conference on Mechatronics (ICM), Takamatsu, Japan, 8–11 August 2021; pp. 1–6. [[CrossRef](#)]

30. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in Real-time. In Proceedings of the Robotics: Science and Systems, Berkeley, CA, USA, 12–26 July 2014. [[CrossRef](#)]
31. Gonzalez, C.; Adams, M. An improved feature extractor for the Lidar Odometry and Mapping (LOAM) algorithm. In Proceedings of the 2019 International Conference on Control, Automation and Information Sciences (ICCAIS), Chengdu, China, 23–26 October 2019; pp. 1–7. [[CrossRef](#)]
32. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural. Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)] [[PubMed](#)]
33. Nagatani, K.; Endo, D.; Yoshida, K. Improvement of the Odometry Accuracy of a Crawler Vehicle with Consideration of Slippage. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 10–14 April 2007; pp. 2752–2757. [[CrossRef](#)]
34. Borenstein, J.; Feng, L. Gyrodometry: A new method for combining data from gyros and odometry in mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 22–28 April 1996; Volume 1; pp. 423–428. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.