sensors

MDPI

Article

Trainable Quaternion Extended Kalman Filter with Multi-Head Attention for Dead Reckoning in Autonomous Ground Vehicles

Gary Milam [1,†], Baijun Xie [1,†], Runnan Liu [1,†], Xiaoheng Zhu [1,†], Juyoun Park [2], Gonwoo Kim [3] and Chung Hyuk Park [1,*]

[1] Department of Biomedical Engineering, George Washington University, Washington, DC 20052, USA
[2] Korea Institute of Science and Technology, Seoul 02792, Korea
[3] Department of Control and Robot Engineering, ChunBuk National University, Chungbuk 28644, Korea
* Correspondence: chpark@gwu.edu
† These authors contributed equally to this work.

Abstract: Extended Kalman filter (EKF) is one of the most widely used Bayesian estimation methods in the optimal control area. Recent works on mobile robot control and transportation systems have applied various EKF methods, especially for localization. However, it is difficult to obtain adequate and reliable process-noise and measurement-noise models due to the complex and dynamic surrounding environments and sensor uncertainty. Generally, the default noise values of the sensors are provided by the manufacturer, but the values may frequently change depending on the environment. Thus, this paper mainly focuses on designing a highly accurate trainable EKF-based localization framework using inertial measurement units (IMUs) for the autonomous ground vehicle (AGV) with dead reckoning, with the goal of fusing it with a laser imaging, detection, and ranging (LiDAR) sensor-based simultaneous localization and mapping (SLAM) estimation for enhancing the performance. Convolution neural networks (CNNs), backward propagation algorithms, and gradient descent methods are implemented in the system to optimize the parameters in our framework. Furthermore, we develop a unique cost function for training the models to improve EKF accuracy. The proposed work is general and applicable to diverse IMU-aided robot localization models.

Keywords: localization; inertial navigation system; extended Kalman filter; mobile robot; autonomous ground vehicle

Citation: Milam, G.; Xie, B.; Liu, R.; Zhu, X.; Park, J.; Kim, G.; Park, C.H. Trainable Quaternion Extended Kalman Filter with Multi-Head Attention for Dead Reckoning in Autonomous Ground Vehicles. *Sensors* **2022**, *22*, 7701. https://doi.org/10.3390/s22207701

Academic Editor: Natividad Duro Carralero

Received: 20 September 2022
Accepted: 7 October 2022
Published: 11 October 2022

1. Introduction

Over the past years, localization has become one of the challenging issues for autonomous ground vehicles (AGVs). In particular, the most difficult issue in navigation is estimating the accurate and stable position and orientation of the robot through data obtained from sensors and other navigation systems [1]. Recently, various technologies have been designed to solve robot localization problems, such as visual-odometry-aided camera localization [2] and Global Positioning System (GPS)-based localization using reinforcement learning [3].

However, the simultaneous localization and mapping (SLAM) methods may fail to function correctly in certain complicated situations due to the physical characteristics of sensors. The laser imaging, detection, and ranging (LiDAR) sensor, for example, is a sensor system that measures distance by transmitting light into spaces and receiving reflected signals from a target [4]. However, LiDAR can lose its signal in situations such as foggy and rainy conditions. In addition to that, the strength of the signal can be affected by the reflectivity of the objects. This could lead a mobile robot or an AGV into a target-blind zone, endangering safety and maneuverability. Therefore, a reliable contingency plan needs to be considered that can compensate for the performance degradation caused by these limitations of the sensors.

The inertial measurement unit (IMU), a combination of a gyroscope, accelerometers, and sometimes magnetometers, could provide an efficient approach to solve this problem. Specifically, the IMU is one of the solutions that can independently measure the state of the body without any external feedback. The accelerometer and gyroscope components measure linear acceleration and angular velocity, which represent the movement of the body to which the sensor is attached. Additionally, the magnetometer component measures orientation based on the Earth's magnetic field, which is available almost everywhere [5].

Nevertheless, there is one drawback that could degrade the performance of an IMU-based localization, which is the accumulated drift. To reliably utilize the IMU, it is imperative to eliminate the accumulated drift [1]. In the field of probabilistic robotics, various methods for correcting errors, such as Bayes filters, Gaussian filters (e.g., information filters), and nonparametric filters (e.g., particle filters) [6], have been proposed. Among them, the extended Kalman filter (EKF) [7] is one of the most widely used methods to reduce the accumulated drift. The EKF is a nonlinear Kalman filter (KF) that linearizes a current mean and covariance estimate. Since EKF can solve nonlinear problems, it has been applied to IMU-aided localization systems [8–10]. The process-noise covariance matrix, Q, and the measurement-noise covariance matrix, R, are constructed with a priori constant values determined by the characteristics of sensors and environments in traditional KF systems, which assume that they remain constant throughout the whole navigation operation. EKF can achieve optimal results if the process noise is well defined. However, depending on external factors, such as complex environments or sensor limitations (e.g., occlusions), sensor noise values can change, and it is difficult to recognize the exact error and in situ information of when and how the change occurs [11].

The following is a list of the major contributions of this study:

1. In this work, we propose a novel approach to improve the accuracy of EKF-based IMU localization with a convolutional neural network (CNN) architecture. Specifically, we design a stable training method that can find the optimal parameters of the system and the observation-noise covariance in real time by reducing the error in each iteration. Furthermore, the system is designed and tested for online training, unlike many other approaches, such as [12], where the algorithm is trained offline using batch and multiple epochs. The intention behind this is that the algorithm is to be trained continuously while SLAM is functioning online, in which case a sequence of IMU data points is observed and acquired.
2. Our proposed CNN module consists of multi-head attention (MHA) layers to model the cross-modal fusion of different sources of modalities (e.g., multiple IMUs, lidars, etc.). The MHA was initially proposed to address the problems of natural language processing (NLP) [13], and it was later discovered to be effective in modeling cross-modal interactions between different modalities [14]. These previous works inspired us to model cross-modal interactions that combine different sensor information sources via the attention mechanism.
3. We conducted extensive experiments using an actual robotic platform to assess the effectiveness of our proposed method in the real world (a factory environment in our case). We designed real-world scenarios for the online training, where the SLAM might fail in some cases and only the IMU(s) can provide sensory information for the EKF-based localization module. The algorithm is also trained continuously while the robot is online and navigating.

2. Related Work

Studies have been conducted recently on Kalman filter (KF)-based localization technology with adaptive noise-covariance estimation. One previous study proposed by Akhlaghi et al. [15] introduced innovation-based and residual-based methods to adaptively adjust the covariance matrices Q and R at each step of the EKF process to improve the state estimation accuracy. In addition, Hu et al. [16] proposed an adaptive unscented Kalman filter (UKF), another variant of KF for the nonlinear system, with process-noise covariance

estimation to improve the UKF performance. However, these approaches might not be able to fully characterize the nonlinear stochastic noises that arise in real-world situations.

It is known that the artificial neural network has the capability to approximate nonlinear functions [17]. Haarnoja et al. [18] demonstrated that a backpropagation-based Kalman filter, consisting of a KF and a CNN, was capable of predicting the measurement-noise covariance matrix, where the CNN was trained via minimizing position errors. Brossard et al. [12] proposed an approach for dead-reckoning for wheeled vehicles with the IMU only. Deep neural networks were used to update the parameters of an invariant EKF dynamically. A recent study also explored the use of long short-term memory (LSTM), a type of recurrent neural network (RNN), to model the nonlinear noises for KF [19] to address target tracking problems. Another approach that uses reinforcement learning to adaptively estimate the process-noise covariance matrix was proposed by Gao et al. [20], in which their algorithm used the deep deterministic policy gradient (DDPG) to extract the optimal process-noise covariance matrix estimation from the continuous action space, using an integrated navigation system as the environment and the reverse of the current positioning error as the reward. Wu et al. [21] also proposed a deep learning framework combining a denoising autoencoder and a multitask temporal CNN. Multitask learning was used to optimize the loss for both the process-noise covariance and measurement-noise covariance matrices from KF simultaneously.

3. Quaternion-Based Extended Kalman Filter

3.1. IMU Inclination Calculation

The rotation matrix $\mathbf{R}_n^b$, mapping the navigation frame n to the body frame b, can be represented by ϕ (rotation angle along the x-axis), θ (rotation angle along the y-axis), and ψ (rotation angle along the z-axis), as follows (trigonometric functions *sin* and *cos* are denoted as s and c, respectively):

$$\mathbf{R}_n^b = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ c\psi s\theta s\phi - c\phi s\psi & c\phi c\psi + s\theta s\phi s\psi & c\theta s\phi \\ c\phi c\psi s\theta + s\phi s\psi & c\phi s\theta s\psi - c\psi s\psi & c\theta c\phi \end{bmatrix}. \tag{1}$$

When the IMU is stationary or moving at a constant speed, the acceleration in the navigation frame should be equal to the gravity constant g, so the inclination angles θ and ϕ can be calculated by Equation (2) as in [22]:

$$\begin{bmatrix} a_x^b \\ a_y^b \\ a_z^b \end{bmatrix} = \mathbf{R}_n^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \Rightarrow \begin{cases} \theta = \arcsin(-\frac{a_x^b}{g}) \\ \phi = \arctan(\frac{a_y^b}{a_z^b}) \end{cases}. \tag{2}$$

The IMU coordinate frame is assumed as shown in Figure 1, where the z-axis is upward, so gravity has a positive value.

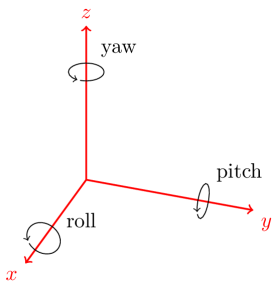

Figure 1. The IMU coordinate frame.

3.2. IMU Integration Model

The continuous-time relationships among position $\mathbf{p}^n$, velocity $\mathbf{v}^n$, and acceleration $\mathbf{a}^n$ are defined as follows:

$$\frac{\partial \mathbf{p}_t^n}{\partial t} = \mathbf{v}_t^n, \ \frac{\partial \mathbf{v}_t^n}{\partial t} = \mathbf{a}_t^n, \tag{3}$$

where $\mathbf{a}_t^n$ represents the acceleration in the navigation frame at time t, which can be calculated using $\mathbf{a}_t^b$, which is the acceleration obtained from the IMU sensor as follows:

$$\mathbf{a}_t^n = \mathbf{R}(\mathbf{q}_b^n)\mathbf{a}_t^b - \mathbf{g}^n, \tag{4}$$

where $\mathbf{R}(\mathbf{q}_b^n)$ is the rotation matrix represented by quaternion $\mathbf{q}_b^n$. The orientation $\mathbf{q}_b^n$ and the angular velocity $\boldsymbol{\omega}^b$ are related as

$$\frac{\partial \mathbf{q}_b^n}{\partial t} = \mathbf{q}_b^n \odot \frac{1}{2}\boldsymbol{\omega}^b. \tag{5}$$

From the continuous-time model, the dynamics of position, velocity, and orientation in discrete time are given by Equations (6)–(8), as explained in [23], as follows:

$$\mathbf{p}_t^n = \mathbf{p}_{t-1}^n + \mathbf{v}_{t-1}^n \cdot \delta t + \frac{1}{2}(\mathbf{a}_{t-1}^n + \mathbf{e}_{a,t}) \cdot \delta t^2 \tag{6}$$

$$\mathbf{v}_t^n = \mathbf{v}_{t-1}^n + (\mathbf{a}_{t-1}^n + \mathbf{e}_{a,t}) \cdot \delta t \tag{7}$$

$$\mathbf{q}_{b,t}^n = \mathbf{q}_{b,t-1}^n \odot exp_q(\frac{1}{2}(\boldsymbol{\omega}_{t-1}^b - \mathbf{e}_{\omega,t}) \cdot \delta t), \tag{8}$$

where $\mathbf{e}_{a,t}$, $\mathbf{e}_{\omega,t}$ are the noise terms of the dynamics model which are assumed to follow the normal distribution, and the distribution axes are independent of each other, as follows:

$$\mathbf{e}_{a,t} \sim \mathcal{N}(0, \sigma_a \mathcal{I}_3) \tag{9}$$

$$\mathbf{e}_{\omega,t} \sim \mathcal{N}(0, \sigma_\omega \mathcal{I}_3). \tag{10}$$

The state variable $\mathbf{x}_t$ is a 10×1 vector consisting of the current position $\mathbf{p}_t^n$, velocity $\mathbf{v}_t^n$, and orientation $\mathbf{q}_{b,t}^n$ which represents the mapping of the body frame onto the navigation frame, as follows:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t^n, & \mathbf{v}_t^n, & \mathbf{q}_{b,t}^n, \end{bmatrix}_{10\times 1}^T \tag{11}$$

so the state transition function can be written as

$$\hat{\mathbf{x}}_{t|t-1} = f(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t, \mathbf{e}_t), \tag{12}$$

where $\mathbf{u}_t = [\mathbf{a}_t^b, \boldsymbol{\omega}_t^b]^T$ is the control input modeled by the accelerometer and gyroscope measurements, and the noise term $\mathbf{e}_t = [\mathbf{e}_{a,t}, \mathbf{e}_{\omega,t}]^T$.

We linearize Equation (12) at the current estimate and propagate the covariance forward to predict the system covariance

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{G}_t \mathbf{Q} \mathbf{G}_t^T, \tag{13}$$

where $\mathbf{F}_t$, $\mathbf{G}_t$ are Jacobian matrices of the transition function (12) with respect to $\mathbf{x}_t$ and $\mathbf{u}_t$, as shown below:

$$\mathbf{F}_t = \left. \frac{\partial f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)}{\partial \mathbf{x}_t} \right|_{\substack{\mathbf{e}_t = 0 \\ \mathbf{x}_t = \hat{\mathbf{x}}_{t-1|t-1}}} \tag{14}$$

$$\mathbf{G}_t = \left. \frac{\partial f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)}{\partial \mathbf{u}_t} \right|_{\substack{\mathbf{e}_t = 0 \\ \mathbf{x}_t = \hat{\mathbf{x}}_{t-1|t-1}}}, \tag{15}$$

and the process covariance $\mathbf{Q} = diag(\Sigma_a, \Sigma_a, \Sigma_\omega)$, where Σ_a and Σ_ω represent the covariances of acceleration and angular velocity, respectively.

3.3. EKF Correction with Measurements

Since the accelerometer measures the local gravity vector when an AGV is moving at a constant speed or is stationary, it can provide information about the inclination of the sensor [23]. Then, the robot control center can provide the velocity information $\mathbf{v}^b$ along the x-axis and y-axis at current times. In addition, we consider the vertical velocity, which is roughly null in the robot frame, as a pseudo-velocity measurement v^b_{pseudo}, so the total measurement vector can be written as

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{a}^b_t, & \mathbf{v}^b_{cmd,t}, & v^b_{pseudo,t} = 0 \end{bmatrix}^T_{6\times 1}. \tag{16}$$

Thus, we can obtain the measurement function mapping the state space to the measurement space as follows:

$$h(\hat{\mathbf{x}}_{t|t-1}) = \begin{bmatrix} \mathbf{R}(\hat{\mathbf{q}}^n_{b,t|t-1})^T \mathbf{g}^n \\ \mathbf{R}(\hat{\mathbf{q}}^n_{b,t|t-1})^T \hat{\mathbf{v}}^n_{t|t-1} \end{bmatrix}, \tag{17}$$

Then, we obtain the measurement matrix $\mathbf{H}_t$ by linearizing the measurement function:

$$\mathbf{H}_t = \left. \frac{\partial h(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right|_{\mathbf{x}_t = \hat{\mathbf{x}}_{t|t-1}}. \tag{18}$$

Therefore, the measurement residual $\mathbf{y}_t$ and the Kalman gain $\mathbf{K}_t$ are calculated by Equations (19) and (20), as detailed in [24]:

$$\mathbf{y}_t = \mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1}) \tag{19}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R})^{-1}, \tag{20}$$

where the measurement covariance matrix $\mathbf{R}$ can be defined as $diag(\Sigma_a, \Sigma_v, \Sigma_{v_{pseudo}})$. Finally, the predicted state and covariance are updated as follows:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\mathbf{y}_t, \tag{21}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1}. \tag{22}$$

4. Covariance Optimization

4.1. Adjustable Covariance

It is well known that the EKF is a model-based optimal filter, which requires exact knowledge of process and measurement models as well as process- and measurement-noise statistics. However, it is difficult to model the dynamic noise changes over time [25]. Thus, we redesign the covariance Σ of the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ as follows:

$$\Sigma_a = \sigma_a^2 \cdot 10^{\mu \tanh(s_a)} \tag{23}$$

$$\Sigma_\omega = \sigma_\omega^2 \cdot 10^{\mu \tanh(s_\omega)} \tag{24}$$

$$\Sigma_{v_{cmd}} = \sigma_{v_{cmd}}^2 \cdot 10^{\mu \tanh(s_{v_{cmd}})} \tag{25}$$

$$\Sigma_{v_{pseudo}} = \sigma_{v_{pseudo}}^2 \cdot 10^{\mu \tanh(\mathbf{s}_{v_{pseudo}})}, \tag{26}$$

where σ_a, σ_ω, $\sigma_{v_{cmd}}$, and $\sigma_{v_{pseudo}}$ correspond to our initial estimate of the noise parameters and $\mu > 0$. Thus, the covariance can be limited between a factor $10^{-\mu}$ and a factor 10^{μ} with respect to its original value because of the function $\tanh(\cdot)$, which makes the covariance

optimized within a reasonable interval set heuristically. By adjusting the value of the parameter s, the covariance matrices could be changed indirectly.

Parameters s_a and s_ω are adjusted during training by backpropagation based on the loss function described by Equation (30). Once the training stops, the parameters are considered fixed for the algorithm.

Although the noise components of the velocity v and pseudo-velocity v_{pseudo} are unknown, the deviation can be assumed to be dynamic rather than stationary in the real world. In other words, the measurement covariance from velocity can be treated as loose strict null instead of strict null, which means that the uncertainty can be encoded in the covariance [12]. A CNN layer is applied to dynamically compute the parameters s_v and $s_{v_{pseudo}}$, taking as input a window size of N IMU data points.

$$\begin{bmatrix} \mathbf{s}_v \\ s_{v_{pseudo}} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} s_{v_x} \\ s_{v_y} \end{bmatrix} \\ s_{v_{pseudo}} \end{bmatrix} = CNN([u_{t-N}, ...u_t]). \tag{27}$$

Figure 2 shows the CNN architecture used to predict parameters s_v and $s_{v_{pseudo}}$. A window size of 20 IMU data points was used for the input. Each IMU data point consists of acceleration and angular velocity data for the three axes (x,y,z). Using a single channel, where the overall input dimension becomes $1 \times 6 \times 20$, the input is initially split into its individual acceleration and angular velocity matrices, which are processed separately by their respective 3×3 convolution layers followed by a leaky ReLU activation (ConvLR block). Then, a multi-head attention (MHA) layer is introduced to model the feature-fusion data. The output of the individual paths is then concatenated and processed by two more ConvLR blocks and a global average pooling layer. The final result is the three parameters s_{v_x}, s_{v_y}, and $s_{v_{pseudo}}$.

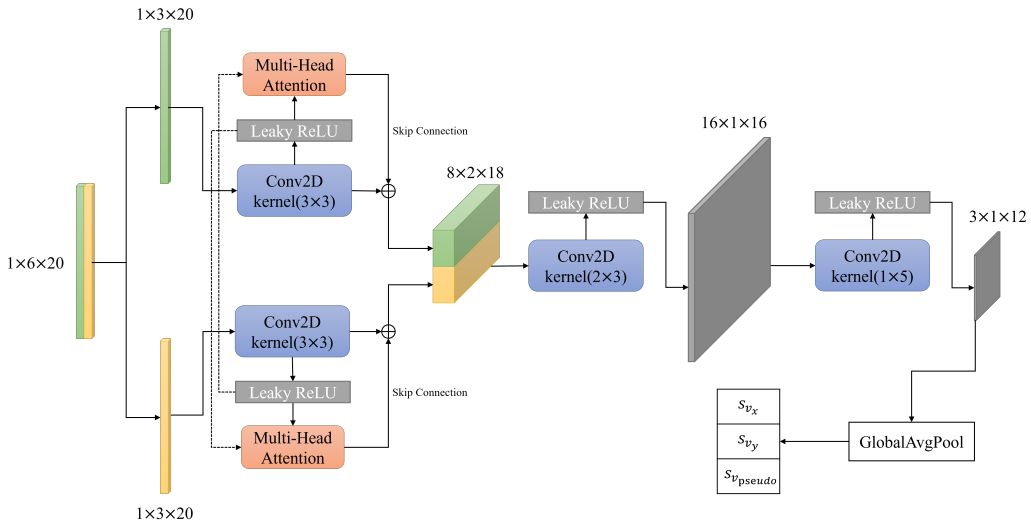

Figure 2. The proposed CNN architecture for predicting the parameters s_{v_x}, s_{v_y}, and $s_{v_{pseudo}}$, which are used to calculate the covariances as per Equations (25) and (26). The input consists of a window size of 20 IMU data points, each containing the acceleration and angular velocity data for all three axes.

The MHA mechanism was initially proposed in the field of natural language processing (NLP) [13]. Later, Tsai et al. [14] explored leveraging MHA mechanism to reinforce a target modality with features from another data modality via learning the cross-modal attention. The following is the formulation of the attention output:

$$\textbf{Attention}(Q_A, K_B, V_B) = \textbf{softmax}(\frac{Q_A K_B^\top}{\sqrt{d_k}}) V_B. \tag{28}$$

Following the definition of [14], $Q_A \in \mathbb{R}^{d \times d_A}$ denotes the queries from the modality A, $K_B \in \mathbb{R}^{d \times d_B}$ denotes the set of keys, and $V_B \in \mathbb{R}^{d \times d_B}$ denotes the set of values from the modality B. Then, the information from modality B is passed to modality A by calculating the attention function in Equation (28). In this study, we leverage the MHA to model feature fusion via introducing one from another modality. As shown in Figure 2, taking acceleration data as an example, we reinforce its features via modeling the cross attention with angular velocity data by calculating attention output $\textbf{Attention}(Q_{acc}, K_{ang}, V_{ang})$. A skip connection is also implemented, to sum up the output from the first 3×3 convolution layers and attention output.

4.2. Online Training Method

SLAM works well unless a sensor fails [26], which is a well-known problem statement. Therefore, our training system initially considers the SLAM outputs under ideal (reliable) conditions as ground truths to calculate the loss function between the output states of SLAM and EKF.

Furthermore, the iterative EKF estimation process in consecutive time steps is a kind of Markov decision process, which means that the current state is only related to the previous one, so our method focuses on optimizing each EKF estimation process. When the estimation performance of each EKF iteration is high, it will show high estimation accuracy in the entire iterative process. In order to evaluate the performance of each EKF estimation, the loss function is designed as follows:

$$loss = \text{MSE}\,(\,\hat{\mathbf{x}}_{t+1|t+1}\,,\,\mathbf{x}_{t+1}^{slam}), \tag{29}$$

where MSE is the mean squared error (MSE) function that expresses the bias of the estimated state by EKF compared to the state of SLAM at timestamp $t+1$. However, the performance of AGV localization depends on the two-dimensional (2D) position errors (p_x, p_y) and the heading angle (ψ) errors, so the loss function only needs to compute the mean squared error of p_x, p_y, and ψ calculated from the orientation state $\mathbf{q}$. Thus, the loss function can be rewritten as

$$loss = \text{MSE}\,(\,[\,\hat{p_x}, \hat{p_y}, \hat{\psi}\,]_{t+1}^{T}\,,\,[\,p_x^{slam}, p_y^{slam}, \psi^{slam}\,]_{t+1}^{T}\,). \tag{30}$$

The initial state of each EKF iteration should be the same as the state of SLAM at the previous timestamp, so that the loss function can effectively express the error generated by each EKF iteration. Therefore, the input of trainable EKF that estimates the state at the next timestamp should be the state from SLAM at the current timestamp during training as follows:

$$\hat{\mathbf{x}}_{t+1|t+1} = \text{EKF}(\mathbf{x}_t^{slam}\,,\,\mathbf{u}_t\,,\,\mathbf{v}_{t+1}^{cmd},\,\mathbf{P}_t^{slam}), \tag{31}$$

which also considers the initial covariance $\mathbf{P}_t$ of EKF as the current estimation covariance from SLAM.

However, the frequency of SLAM is different from the frequency of IMU in real-time, so we cannot guarantee that the estimated state of each EKF iteration corresponds to the output state of each SLAM at the same time, which means that it is unable to regard the ground truth of EKF as SLAM at that time. Additionally, the frequency of IMU is usually higher than the frequency of SLAM, so to synchronize the output of EKF and SLAM, multiple EKF iterations can be trained by performing backpropagation [27] at once when the output state of SLAM is provided. The training structure is shown in Figure 3. The structure allows the system to optimize the covariance in real-time continuously. The training process begins with the MSE loss function (30), and then calculates the gradient of the loss function corresponding to each $\mathbf{Q}$ and $\mathbf{R}$ based on the derivative chain rule.

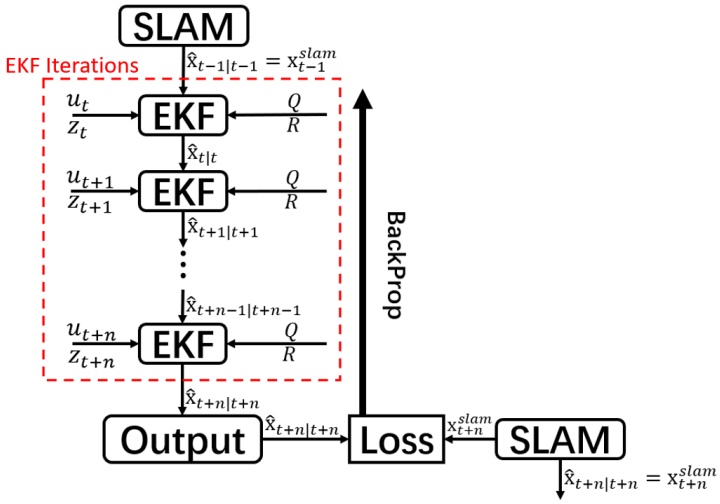

Figure 3. Structure and training flow of our network. The initial state of the EKF iterations is based on the previous SLAM state update at timestamp $t-1$. N number of EKF iterations will be performed until the next update of the SLAM state at $t+n$. The loss is calculated between the SLAM state update and the final predicted output state of the N EKF updates. Then, backpropagation is performed on the basis of the calculated loss.

4.3. Implementation Details

This section introduces the settings and the implementation details of our algorithm. The whole self-adjusting method is implemented in Python with the PyTorch library [28] for training and inference, and the Robot Operating System (ROS) [29] was utilized for collecting sensor data.

The initial parameters of the EKF were set as follows prior to training. The initial system error covariance $\mathbf{P}_0 = \mathbf{I}_{10}$, which is the identity matrix 10×10. We set $\sigma_a = 0.01$ m/s^2 in (23), $\sigma_\omega = 0.01$ rad/s^2 in (24), $\sigma_v = 0.25$ m/s in (25), and $\sigma_{v_{pseudo}} = 0.0225$ m/s in (26). The adjustable parameters s are defined as $\mathbf{s}_a = \mathbf{s}_\omega = \mathbf{s}_{v_{cmd}} = s_{v_{pseudo}} = 0.01$ in order to make initial covariance $\Sigma_a \approx \sigma_a^2$, $\Sigma_\omega \approx \sigma_\omega^2$, $\Sigma_{v_{cmd}} \approx \sigma_{v_{cmd}}^2$, and $\Sigma_{v_{pseudo}} \approx \sigma_{v_{pseudo}}^2$. We defined $\mu = 3$ from (23) to (26), which allows for each covariance element to be 10^3 times higher or smaller than its original values [12].

The Adam optimizer [30] was applied to update parameters with learning rate 10^{-3} and iterated once for each tandem EKF.

5. Experiments and Results

In this section, we present the robotic platform that was used for collecting data and testing a real-time SLAM failure scenario. Three evaluation metrics were used for evaluating the performance. The performance of our proposed localization architecture, the trainable quaternion-based EKF, was evaluated by comparing it with the EKF model without training. Additionally, various deep learning model architectures were compared to determine the most effective model.

5.1. Dataset

The data collected to evaluate the proposed algorithm are based on the usage of a proprietary omniwheel robot platform of dimensions of 2.481×1.595 meters. The robotic system provides the current velocity of the robot, which is used as an input to the EK. The robot contains four 2D LiDARs (two in the front and two in the rear), two stereo cameras (one in the front and one in the rear), and an IMU. Figure 4 shows the robot setup.

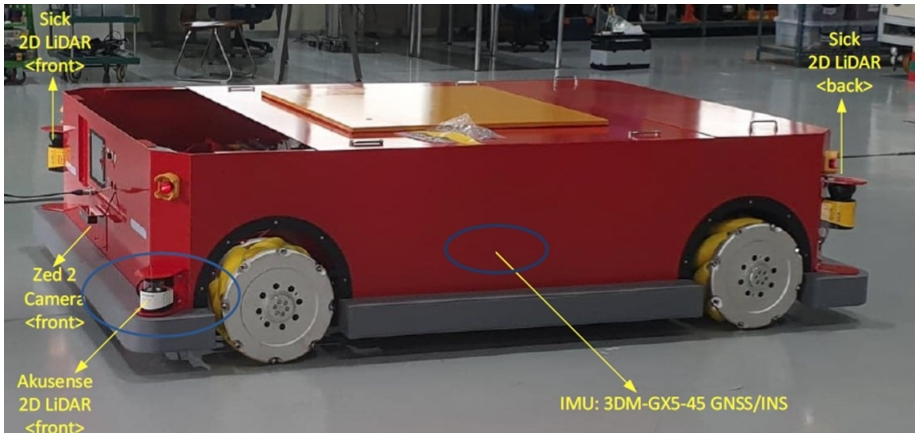

Figure 4. Robot setup used for data collection of the experiment.

Experiments were conducted across four different trajectories with total lengths of 66.46, 145.39, 103.42, and 78.62 meters, respectively. The trajectory path collected as a result of SLAM was used as the ground truth. For this work, the visual–LiDAR–inertial SLAM algorithm in [31] was used, as it provides a very accurate position and orientation. The dataset collected contained the SLAM output (ground truth position and orientation), the acceleration and angular velocity information, and IMU data for the EKF state estimation calculations.

Since the EKF state estimation was calculated at the origin of the IMU, all data points (SLAM and velocity) were transformed to the IMU's origin. Furthermore, all data had to be transformed to the body frame coordinates of the omniwheel robot, where the x-axis runs along the length of the robot in the forward direction, the y-axis is 90 degrees anticlockwise, and the z-axis runs upward.

5.2. Experimental Setup

To evaluate the improvement of the proposed algorithm, the localization output of a traditional EKF was used as the experimental control data.

Our proposed system architecture was designed with the intention that the algorithm would be trained continuously while the robot is online; therefore, the data are collected in real time, and each data point is unique and independent. Taking this into account, we trained the algorithm on three sequence runs and performed the inference test on the fourth unseen sequence, with the data of each sequence only seen once during training and inference. This process was performed four times to run the inference test on all available sequences.

The EKF state estimation was then fused with the SLAM state during the inference mode while the system considered the SLAM positional data reliable, which is determined by the covariance values provided by the SLAM module. The time range when the SLAM positional data were unreliable was considered the SLAM failure period, at which point the SLAM state was no longer fused with the EKF estimation, and the output was purely relying on EKF.

To simulate a real-time SLAM failure scenario, 100 s time intervals within each individual trajectory were selected as the SLAM failure period. Both approaches, the traditional EKF and the proposed trainable EKF, had the same start and end times for the SLAM failure.

In Figures 5 and 6, the period from when SLAM failed ("Start Point") to when SLAM recovered ("End Point") were regarded as the SLAM failure period. The green and orange paths represent the estimated position from the untrained EKF and EKF CNN with MHA, respectively, while the blue path (ground truth) is from the SLAM outputs.

To evaluate the performance of our proposed algorithms, we considered the following three evaluation metrics for our experiments:

1. *Position Error* (P_e) :
   The ratio of the position error to the total path length when SLAM fails.

$$P_e = (Last\ Position\ Error / Total\ Distance) * 100 \tag{32}$$

2. *Rotation Error* (A_e) :
   Azimuth angle error relative to the total path length when SLAM fails.

$$A_e = Last\ Azimuth\ Angle\ Error / Total\ Distance \tag{33}$$

3. *Average Mean Squared Error* (MSE_{avg}):
   The overall average of the squares of the errors between ground truths (SLAM) and object to be assessed (estimated position and orientation) when SLAM fails.

$$MSE_{avg} = Mean(\ MSE\ (\ [\ \hat{p_x}, \hat{p_y}, \hat{\psi}\ ]_{t+1}^{T}, [\ p_x^{slam}, p_y^{slam}, \psi^{slam}\ ]_{t+1}^{T}\ )\ ) \tag{34}$$

5.3. Result and Analysis

We present the comparison results of our proposed algorithm with those of the traditional method (i.e., EKF). From the trajectory comparisons in Figures 5 and 6, we can see that the proposed EKF CNN with MHA (orange path) was more in line overall with less noise than the traditional EKF (green path). Furthermore, from Figures 7 and 8, which provide the mean squared errors (MSE) during the failure period, we can see that the proposed method (green line) could achieve a lower MSE than the traditional EKF (blue line). From Figures 7 and 8, the maximum MSE for the proposed method was 0.0297 and 0.0323, respectively, while the traditional EKF method reached 0.0690 and 0.1108 for the same trajectories.

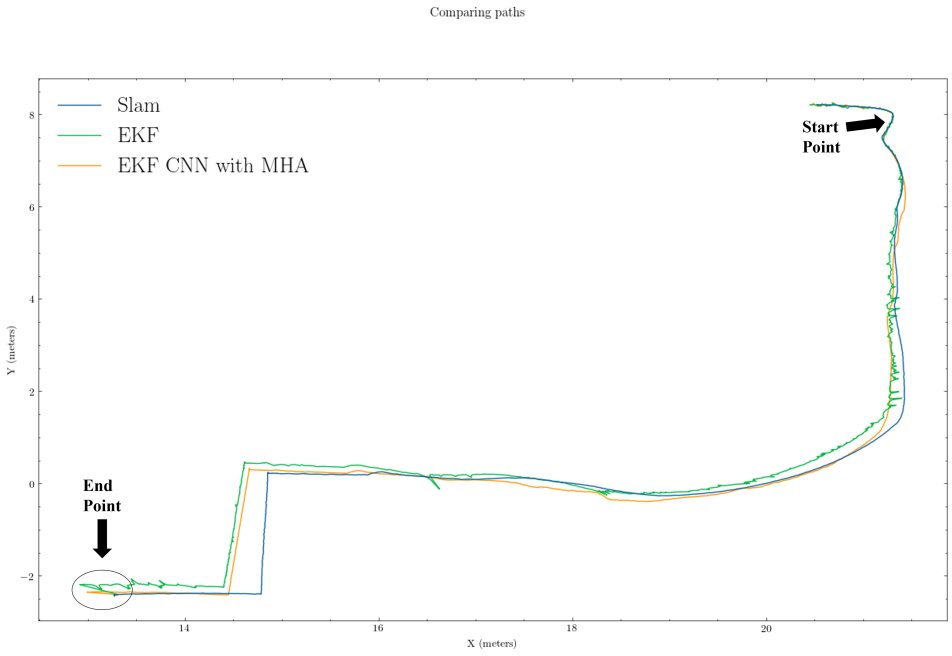

Figure 5. Estimated path comparison results for trajectory 1. Green, orange, and blue lines represent EKF, EKF+CNN with MHA, and ground truths, respectively.

Table 1 shows the full test results we collected and analyzed. As mentioned for each trajectory, an arbitrary interval of 100 s was chosen as the failure period, which could result in different failure lengths. Based on the provided results, it is clear that our proposed method overall outperformed the traditional EKF. In all cases, the position error P_e and the rotation error A_e were lower with the proposed method. In fact, the proposed method

achieved 1.24765% for overall P_e and 0.02785 deg/m for A_e. Similarly, the proposed method performed much better on the MSE metric.

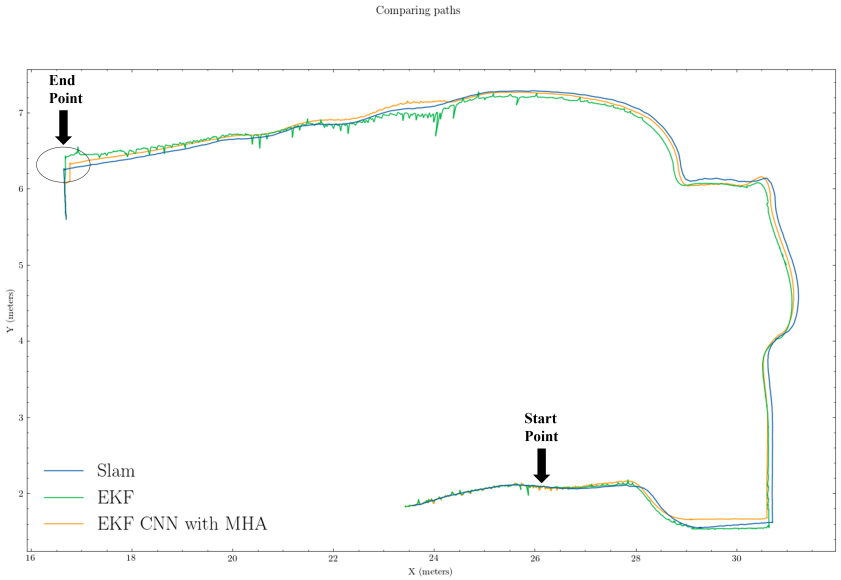

Figure 6. Estimated path comparison results for trajectory 4. Green, orange, and blue lines represent EKF, EKF+CNN with MHA, and ground truths, respectively.

Table 1. Results for the four trajectory paths followed by evaluation metrics. Failure length: moving track length of robot during SLAM failure period; P_e: position error; A_e: rotation error; MSE_{avg}: average mean squared error.

Trajectory	Failure Length (m)	EKF			Proposed		
		P_e (%)	A_e (deg/m)	MSE_{avg}	P_e (%)	A_e (deg/m)	MSE_{avg}
1	18.19	1.64180	0.01320	0.02055	0.97901	0.00639	0.01580
2	23.92	3.56478	0.10087	0.11934	2.79777	0.04045	0.07454
3	22.94	1.00380	0.22513	0.04406	0.84346	0.01948	0.01294
4	24.81	0.38022	0.02335	0.00504	0.37035	0.04509	0.00523
Overall		1.64765	0.09064	0.04725	1.24765	0.02785	0.02713

Our trainable EKF CNN with MHA showed improved results over the traditional EKF estimation due to optimization of process and measurement errors through a combination of CNN inference, backpropagation, and gradient descent.

In Table 2, we also compared the performance of using different model architectures. For the implementation of EKF+CNN_1, we removed the MHA layer with skip connection and kept all the other components, and the EKF+CNN_2 was implemented by removing the last convolution layer of EKF+CNN_1. EKF+LSTM was also introduced to attempt to learn the temporal features from the data. As can be seen, eliminating the final convolution layer reduced *Pe* and *Ae* performance overall while somewhat lowering MSE, which EKF+LSTM also achieved. Nevertheless, the proposed architecture combining CNN and MHA showed considerable improvements in MSE and lowered P_e a bit. The A_e was still higher than that of EKF+CNN_1, but in the acceptable range.

Table 2. Results for the four trajectory paths followed by evaluation metrics. Failure length: moving track length of robot during SLAM failure period; P_e: position error; A_e: rotation error; MSE_o: overall mean squared error. CNN_1 is the proposed model architecture without the multi-head attention layer. CNN_2 is the same as CNN_1 but removes the last convolution layer. LSTM is implemented with a number of 2 layers and a hidden size of 256.

Model	P_e (%)	A_e (deg/m)	MSE_o
EKF	1.64765	0.09064	0.04725
EKF+CNN_1	1.25114	0.02111	0.04084
EKF+CNN_2	1.46609	0.03109	0.03696
EKF+LSTM	1.35288	0.03138	0.03573
Proposed	1.24765	0.02785	0.02713

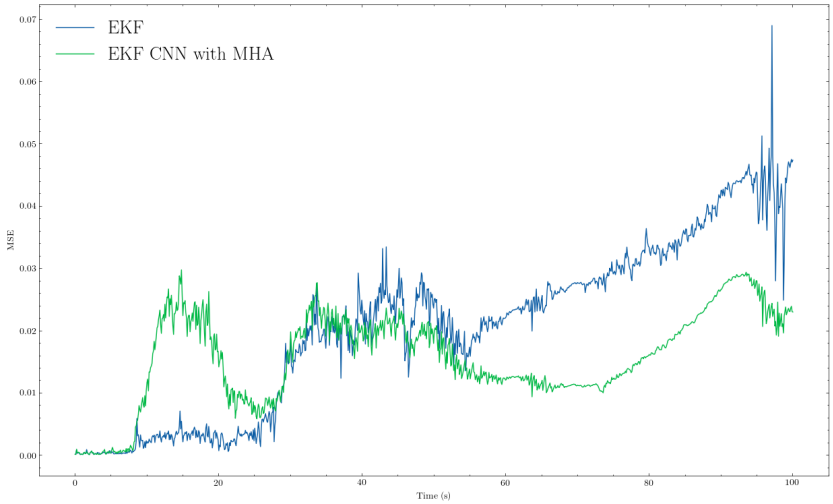

Figure 7. The Mean Squared Error (MSE) comparison between EKF and EKF+CNN with MHA for trajectory 1. Blue and green lines represent EKF and proposed method, respectively

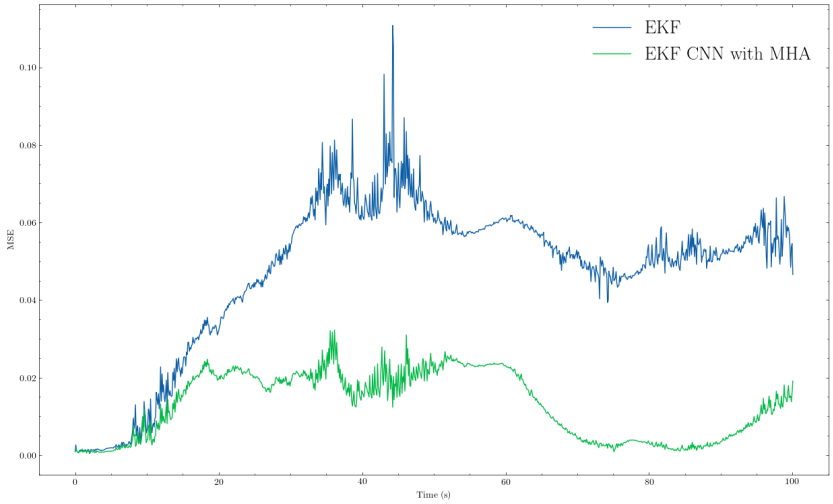

Figure 8. The Mean Squared Error (MSE) comparison between EKF and EKF+CNN with MHA for trajectory 3. Blue and green lines represent EKF and proposed method, respectively.

5.4. Discussion

We demonstrated the advantages of our proposed trainable quaternion-based EKF with CNN and MHA based on the conducted experiments and analysis. Our proposed method could achieve 1.24765% in P_e, 0.02785 in A_e, and it significantly outperforms other model architectures in terms of MSE with 0.02713. The results also suggest that a trainable EKF, which can dynamically adjust process and measurement noise covariance matrices, can improve localization performance. Moreover, our proposed model architecture provides scope for fusing the inputs via reinforcing one modality by introducing features from another modality for the inputs of estimating covariance matrices' parameters for EKF.

However, the online training cost of the proposed method can limit the overall performance. Furthermore, the time lag introduced by the online model training can be accumulated and cause deviations at the endpoints because of the limitations of the embedded computing system. Nonetheless, this issue can be solved by transmitting data to a more powerful server for online training. Furthermore, the future development of the AI-embedded platform will provide more power to achieve better online training performance. In conclusion, our results demonstrate that the deep learning model can be trained and provide predictions in an online training manner in a local integrated system.

6. Conclusions

In this paper, we designed a trainable and adjustable quaternion-based EKF algorithm with CNN and MHA for the sensor fusion of IMU-based localization and visual–LiDAR–inertial SLAM. Specifically, we developed an approach where the EKF-based localization system can provide a more accurate position estimation when SLAM failure occurs during a short time period. This was performed by tuning the process- and measurement-covariance matrices trained by CNN through backpropagation and further adjusting the velocity measurement covariances according to real-time IMU data through network inference. The approach leveraged the SLAM data as ground truths to compute the mean squared error of position and orientation estimated by the EKF while training.

For training and estimation, we designed a tandem EKF structure to adapt to the situation where real-time data from different sources were fused at different frequencies. Our proposed trainable EKF will be effective in dead-reckoning as a complementary process of SLAM when SLAM fails, which will enhance accuracy and stability in localization in complex and dynamic environments. Our future steps will focus on enhancing the structure of our proposed EKF CNN with MHA by performing multi-IMU fusion with multiple EKF modules, where each EKF leverages a unique IMU source.

Author Contributions: Conceptualization, G.M., B.X., R.L., X.Z., J.P., and C.H.P.; methodology, G.M., B.X., R.L., X.Z., and C.H.P.; software, G.M., R.L., X.Z., and J.P.; validation, G.M., B.X., R.L., and X.Z.; formal analysis, G.M. and B.X.; investigation, G.M., B.X., R.L., and X.Z.; resources, G.K. and C.H.P.; data curation, G.M. and B.X.; writing—original draft preparation, R.L., X.Z., and C.H.P.; writing—second draft with additional results and editing, G.M., B.X., and C.H.P.; visualization, J.P., G.K., and C.H.P.; supervision, J.P. and C.H.P.; project administration, G.K. and C.H.P.; funding acquisition, G.K. and C.H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by the Ministry of Trade, Industry and Energy (MOTIE) and the Korea Institute of Advancement of Technology (KIAT) through the International Cooperative R&D program (Project No. P0004631).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Alatise, M.B.; Hancke, G.P. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access* **2020**, *8*, 39830–39846. [CrossRef]
2. Ott, F.; Feigl, T.; Loffler, C.; Mutschler, C. ViPR: Visual-Odometry-aided Pose Regression for 6DoF Camera Localization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Virtual, 14–19 June 2020; pp. 42–43.
3. Zhang, E.; Masoud, N. Increasing GPS Localization Accuracy With Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 2615–2626. [CrossRef]
4. Lidow, A.; De Rooij, M.; Strydom, J.; Reusch, D.; Glaser, J. *GaN Transistors for Efficient Power Conversion*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
5. Zhao, J. A Review of Wearable IMU (Inertial-Measurement-Unit)-based Pose Estimation and Drift Reduction Technologies. *J. Physics: Conf. Ser.* **2018**, *1087*, 9. [CrossRef]
6. Thrun, S.; Burgard, W.; Fox, D. *Probability Robotics*; MIT Press: Cambridge, MA, USA, 2005.
7. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
8. Malyavej, V.; Kumkeaw, W.; Aorpimai, M. Indoor robot localization by RSSI/IMU sensor fusion. In Proceedings of the 2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Krabi, Thailand, 15–17 May 2013; pp. 1–6.
9. Brossard, M.; Bonnabel, S. Learning wheel odometry and IMU errors for localization. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 291–297.
10. Yan, Y.; Zhang, B.; Zhou, J.; Zhang, Y.; Liu, X. Real-Time Localization and Mapping Utilizing Multi-Sensor Fusion and Visual–IMU–Wheel Odometry for Agricultural Robots in Unstructured, Dynamic and GPS-Denied Greenhouse Environments. *Agronomy* **2022**, *12*, 1740. [CrossRef]
11. Jurado, J.; Kabban, C.M.S.; Raquet, J. A regression-based methodology to improve estimation of inertial sensor errors using Allan variance data. *Navig. J. Inst. Navig.* **2019**, *66*, 251–263. [CrossRef]
12. Brossard, M.; Barrau, A.; Bonnabel, S. AI-IMU dead-reckoning. *IEEE Trans. Intell. Veh.* **2020**, *5*, 585–595. [CrossRef]
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
14. Tsai, Y.H.H.; Bai, S.; Liang, P.P.; Kolter, J.Z.; Morency, L.P.; Salakhutdinov, R. Multimodal transformer for unaligned multimodal language sequences. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Volume 2019, p. 6558.
15. Akhlaghi, S.; Zhou, N.; Huang, Z. Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation. In Proceedings of the 2017 IEEE Power & Energy Society General Meeting, Chicago, IL, USA, 16–20 July 2017; pp. 1–5.
16. Hu, G.; Gao, B.; Zhong, Y.; Gu, C. Unscented kalman filter with process noise covariance estimation for vehicular ins/gps integration system. *Inf. Fusion* **2020**, *64*, 194–204. [CrossRef]
17. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
18. Haarnoja, T.; Ajay, A.; Levine, S.; Abbeel, P. Backprop kf: Learning discriminative deterministic state estimators. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 4376–4384.
19. Song, F.; Li, Y.; Cheng, W.; Dong, L.; Li, M.; Li, J. An Improved Kalman Filter Based on Long Short-Memory Recurrent Neural Network for Nonlinear Radar Target Tracking. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 8280428. [CrossRef]
20. Gao, X.; Luo, H.; Ning, B.; Zhao, F.; Bao, L.; Gong, Y.; Xiao, Y.; Jiang, J. RL-AKF: An adaptive kalman filter navigation algorithm based on reinforcement learning for ground vehicles. *Remote Sens.* **2020**, *12*, 1704. [CrossRef]
21. Wu, F.; Luo, H.; Jia, H.; Zhao, F.; Xiao, Y.; Gao, X. Predicting the noise covariance with a multitask learning model for Kalman filter-based GNSS/INS integrated navigation. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 1–13. [CrossRef]
22. Feng, K.; Li, J.; Zhang, X.; Shen, C.; Bi, Y.; Zheng, T.; Liu, J. A new quaternion-based Kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm. *Sensors* **2017**, *17*, 2146. [CrossRef] [PubMed]
23. Kok, M.; Hol, J.D.; Schön, T.B. Using inertial sensors for position and orientation estimation. *arXiv* **2017**, arXiv:1704.06053.
24. Mochnac, J.; Marchevsky, S.; Kocan, P. Bayesian filtering techniques: Kalman and extended Kalman filter basics. In Proceedings of the 2009 19th International Conference Radioelektronika, Bratislava, Slovakia, 22–23 April 2009; pp. 119–122.
25. NGOC, T.T.; KHENCHAF, A.; COMBLET, F. Evaluating Process and Measurement Noise in Extended Kalman Filter for GNSS Position Accuracy. In Proceedings of the 2019 13th European Conference on Antennas and Propagation (EuCAP), Krakow, Poland, 31 March–5 April 2019; pp. 1–5.
26. Khairuddin, A.R.; Talib, M.S.; Haron, H. Review on simultaneous localization and mapping (SLAM). In Proceedings of the 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 27–29 November 2015; pp. 85–90.
27. LeCun, Y.; Touresky, D.; Hinton, G.; Sejnowski, T. A theoretical framework for back-propagation. In Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA, USA, 17–26 June 1988; Volume 1, pp. 21–28.
28. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, 32, 8026–8037.

29. Stanford Artificial Intelligence Laboratory. Robotic Operating System. Available online: https://www.ros.org (accessed on 1 September 2020).
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980
31. Nam, D.V. Robust Multi-Sensor Fusion-based SLAM using State Estimation by Learning Observation Model. Ph.D. Thesis, Chungbuk National University, Cheongju, Korea, 2022.