machines

MDPI

Article

Low-Cost Real-Time Localisation for Agricultural Robots in Unstructured Farm Environments

Chongxiao Liu and Bao Kha Nguyen *

School of Engineering and Informatics, University of Sussex, Brighton BN1 9QT, UK; cl743@sussex.ac.uk
* Correspondence: b.k.nguyen@sussex.ac.uk

Abstract: Agricultural robots have demonstrated significant potential in enhancing farm operational efficiency and reducing manual labour. However, unstructured and complex farm environments present challenges to the precise localisation and navigation of robots in real time. Furthermore, the high costs of navigation systems in agricultural robots hinder their widespread adoption in cost-sensitive agricultural sectors. This study compared two localisation methods that use the Error State Kalman Filter (ESKF) to integrate data from wheel odometry, a low-cost inertial measurement unit (IMU), a low-cost real-time kinematic global navigation satellite system (RTK-GNSS) and the LiDAR-Inertial Odometry via Smoothing and Mapping (LIO-SAM) algorithm using a low-cost IMU and RoboSense 16-channel LiDAR sensor. These two methods were tested on unstructured farm environments for the first time in this study. Experiment results show that the ESKF sensor fusion method without a LiDAR sensor could save 36% of the cost compared to the method that used the LIO-SAM algorithm while maintaining high accuracy for farming applications.

Keywords: multi-sensor fusion; localisation; RTK (real-time kinematic); GNSS (global navigation satellite system); wheel odometry; IMU; unstructured farms; LiDAR; SLAM

Citation: Liu, C.; Nguyen, B.K. Low-Cost Real-Time Localisation for Agricultural Robots in Unstructured Farm Environments. *Machines* **2024**, *12*, 612. https://doi.org/10.3390/machines12090612

Academic Editor: Dan Zhang

Received: 11 August 2024
Accepted: 27 August 2024
Published: 2 September 2024

1. Introduction

The development of agricultural robotics and precision farming techniques has the potential to address labour shortages and enhance production efficiency by significantly reducing the need for manual labour in demanding agrarian tasks. This marks a critical step towards modernising farming practices.

However, unstructured farm environments present unique challenges for agricultural robots. These environments are characterised by their dynamic and unpredictable nature, contrasting with the controlled conditions of greenhouses or structured plantations. Unstructured farms are often marked by irregular terrain, a diversity of crop types, and numerous natural obstacles, including rocks, trees, and water, all contributing to the complexity of navigating and executing tasks. Moreover, the lack of predefined paths and the continuous variation in environmental conditions, including soil moisture levels and plant growth stages, further amplifies the operational demands placed on robotic systems. Such complexities highlight the critical need for advanced sensing, localisation, and navigation capabilities in agricultural robots to effectively adapt to the ever-changing and varied agricultural landscape.

Traditional agricultural robots normally use RTK-GNSS, which provides high-precision positioning. However, these robots often struggle to provide real-time and precise performance in unstructured farms and complex settings [1]. The primary reason is that dense foliage, obstacles, and hills can cause signal attenuation and reflection in most unstructured farm environments. In severe cases, obstacles can reduce the number of visible satellites, rendering the GNSS receiver entirely ineffective [2]. The use of RTK-GNSS for agricultural robots also increases their cost, as the cost of the RTK-GNSS itself can be as high as USD 30,000.

Therefore, the cost has to be lowered to promote the use of high-accuracy localisation in agricultural robotics [3,4]. To address this issue, the multi-sensor fusion approach is often used to incorporate devices such as wheel odometers along with GNSS or to combine GNSS with IMU [5–10]. Lin et al. combined Oriented FAST. They used Rotated BRIEF SLAM (ORB-SLAM), IMU, and wheel odometry to overcome limitations caused by the lack of GPS [11,12].

Tixiao Shan proposed the LiDAR Inertial Odometry via Smoothing and Mapping (LIO-SAM) framework, a SLAM method that couples LiDAR and IMU data through factor graph optimisation. This integration enables high-precision localisation and mapping services in complex environments, and the factor graph optimisation framework enhances the algorithm's robustness and accuracy [13]. However, the algorithm requires an IMU with a high update frequency as well as a high-performance LiDAR, which costs up to GBP 3280 [14]. Although it has excellent technical performance, these cost factors limit its application in agricultural applications.

This study tests and compares two different localisation methods that integrate data from low-cost sensors for real-time localisation in unstructured farm environments. By comparing these two approaches, the performance and feasibility of the low-cost localisation scheme can be evaluated to ensure that the proposed approach meets the needs of real-world applications for agricultural robots. Additionally, the LIO-SAM algorithm is also used to construct a 3D environmental map in this paper.

The primary contributions of this paper are as follows:

1. Use a low-cost, high-accuracy differential localisation system on the robot platform.
2. Develop a low-cost localisation method for unstructured agricultural environments based on ESKF, integrating IMU, RTK-GNSS, and wheel odometry.
3. Test the LIO-SAM algorithm on the agricultural environment for the first time.
4. Adapt the LIO-SAM algorithm to suit the RoboSense RS16 LiDAR and a low-cost IMU for agricultural applications.

The remainder of this paper is organised as follows. Section 2 describes the methods and technologies used in this research. Section 3 presents the experimental design and displays the results obtained. Section 4 analyses and discusses the experimental results. Section 5 concludes the paper.

2. Methodologies

This study proposes an approach to achieve low-cost, real-time localisation and mapping for agricultural robots in unstructured farming environments. It introduces a fusion algorithm integrating data from wheel odometry, IMU, and RTK-GNSS. Among these sensors, wheel odometry can provide short-term accurate motion data, but it is prone to cumulative errors; the IMU offers continuous acceleration and attitude information, but it is limited by drift issues, and RTK-GNSS delivers high-precision localisation data that can be affected by environmental obstacles. The ESKF algorithm is therefore utilised to integrate the data from these sensors and enhance the accuracy of the localisation system while maintaining cost efficiency in this paper. The performance of this method is also compared with the LIO-SAM method.

Figure 1a illustrates the diagram of the low-cost localisation algorithm for fusing data from three sensors using ESKF and the localisation framework diagram of the LIO-SAM algorithm. Initially, localisation data are obtained through the RTK-GNSS system and converted into odometry-format pose information. Then, this information, along with data from the IMU and wheel odometry, is fused using the ESKF to produce the final fused pose information. Figure 1b shows the localisation framework diagram of the LIO-SAM algorithm. It obtains data from the IMU and 3D LiDAR to achieve positioning. The following sections detail the robot platform used in this study, the application of RTK technology, and the specific implementation of the ESKF fusion algorithm and the LIO-SAM algorithm.

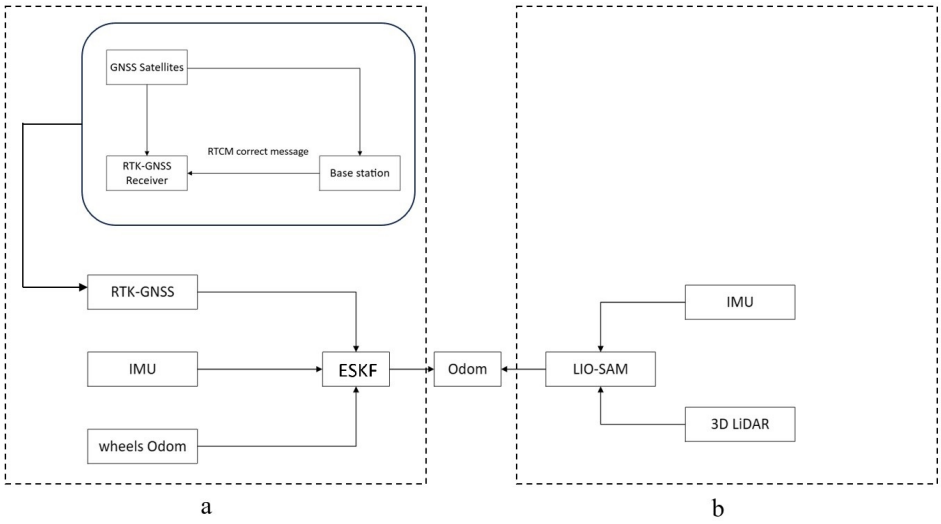

Figure 1. Robot localisation algorithm architecture diagrams using: (**a**) the ESKF-based method. (**b**) the LIO-SAM method.

2.1. Robot Platform

The robot platform used in this research is shown in Figure 2. It uses a six-wheel differential suspension system; the wheels use high-power DC motors to provide sufficient power for the robot to climb and cross the obstacles, and the suspension structure allows the robot to cross the obstacles on the farm smoothly. The robot does not need to operate at high speeds, so a single antenna was used to reduce the cost.

This robot is also equipped with a 16-line LiDAR, which can construct a real-time 3D map of the surrounding environment through the LIO-SAM algorithm. The controller of this robot is NVIDIA's newly released Jetson Orin Nano embedded development board, which provides sufficient arithmetic power for the robot to deal with the computational task of processing and fusing multiple sensors while controlling the cost. Compared to the IMU and LiDAR used in the original version of LIO-SAM, this robot platform would save up to 64% of costs [13].

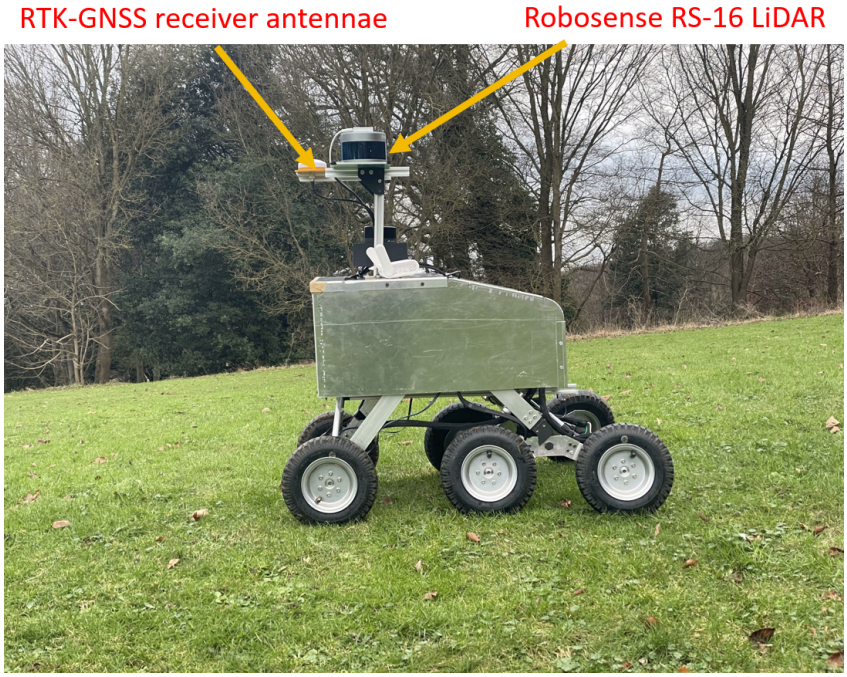

Figure 2. Agriculture robot platform used for this study.

Table 1 shows the price list of navigation sensors used in this study. Figure 3 illustrates the robot's hardware architecture, showing how it perceives its environment using LiDAR, an IMU, and RTK-GNSS. Data from these sensors are transmitted to the main controller through serial and Ethernet connections. An STM32 controller is used to control the robot's movements.

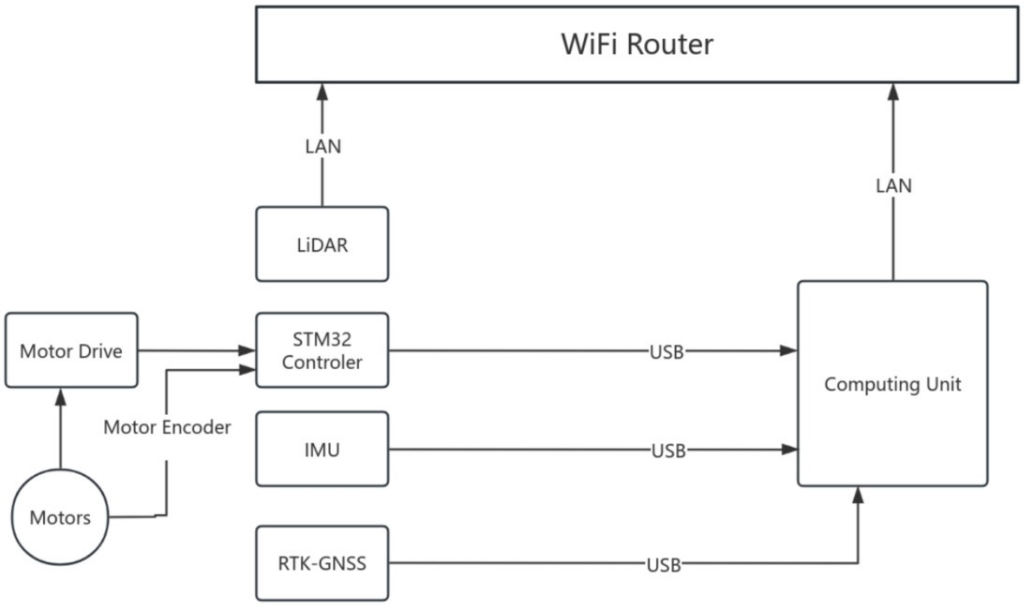

Figure 3. Hardware connection diagram for the robot used in this study.

The robot platform dimensions are 800 mm × 500 mm × 430 mm, weighing 20 kg and having a load capacity of 10 kg. Designed to work on the complex and variable terrain of farms, the robot is equipped with a 6-wheel suspension system, and the minimum ground clearance is 300 mm. This design enables the robot to overcome obstacles efficiently and ensures stable operation on slopes up to 30°. The robot's maximum moving speed is 0.5 m/s.

The LiDAR system on this robotic platform employs the low-cost RS-LiDAR-16, a 16-channel LiDAR from RoboSense. It measures from 0.4 m to 150 m with an accuracy of ±2 cm. Field of view is 30°, and the angular resolution is 0.1°. This LiDAR operates at a rotational speed of 1200 rpm, enabling a sampling frequency of 20 Hz. Point cloud data are transmitted to the robot controller via a 100 Mbps Ethernet port. High-frequency sampling is crucial for navigation and mapping in unstructured farm environments. The LiDAR operates at a wide range of temperatures, from −30 °C to 60 °C, which allows the robot to operate in complex climatic conditions on farms.

This robot platform utilises two u-blox ZED-F9P modules to establish a high-precision differential localisation system for localisation. One module is a rover fixed on the robot, and the other module is a base station fixed on the ground. It receives signals from the Global Navigation Satellite System (GNSS) as well as differential data from a base station. By using differential positioning technology, it achieves an accuracy of up to 14 mm.

The low-cost Wheeltec N100 IMU is used as its primary inertial measurement unit. The N100 IMU provides a pitch and roll accuracy of 0.05° RMS, which is crucial for precise motion control and stable navigation. It features a high output frequency of up to 200 Hz, ensuring the robot system can respond quickly and accurately to environmental changes.

Table 1. List of sensor expenses used in this study.

Components	Model and Specifications	Quantity	Unit Price	Total Price
RTK Module	ZED F9P Module	2	GBP 110	GBP 220
IMU	Wheeltec N100	1	GBP 29	GBP 29
RTK Antenna		1	GBP 39	GBP 39
		Total for ESKF-based Method		GBP 288
3D LiDAR	RS-16	1	GBP 670	GBP 670
IMU	Wheeltec N100	1	GBP 29	GBP 29
		Total for LIO-SAM Method		GBP 699

2.2. Real-Time Kinematic RTK

The robot kinematic model provides the robot's relative position by using data from its internal sensors, such as the wheel odometry and IMU. This model helps understand the robot's movements in terms of its frame of reference, estimating changes in position and orientation as the robot moves. However, the kinematic model alone cannot determine the robot's absolute position in a real-world coordinate system because it lacks global positioning information.

In unstructured farm environments, RTK techniques can be used to obtain the robot's coordinates in the real world, as discussed in this section. These techniques provide absolute positioning data that can be combined with the relative position estimates from the kinematic model. By integrating data from RTK-GNSS with the robot's kinematic model through ESKF, the robot can accurately map its position within a larger, real-world framework.

GNSS (global navigation satellite system) provides global location data (latitude, longitude, altitude) [15]. The most recognised system is the US GPS, offering worldwide localisation and time data. Russia's GLONASS, similar to GPS, serves internationally. The EU's newer system Galileo aims for precise European services. China's BeiDou, initially regional, has recently achieved global coverage.

In RTK systems, a base station is often used, and a rover or receiver is placed on a mobile robot. The base station, located at a surveyed position, can estimate the errors for each GNSS signal received. Once these error corrections are transmitted to the user's receiver, integer ambiguity resolution (IAR) computations are carried out. This principle is more suitable when the user is relatively close to the reference station. However, if the distance between the user and the reference station is significant, atmospheric conditions at the two locations might differ. This disparity can lead to the unsuccessful execution of IAR operations. The typical guideline for maximum distance is around 25 km.

Another approach is Network RTK, which requires network access at the receiver's location. In this method, the receiver, acting as a client receives differential signals from network stations [16,17]. This model is not limited by coverage area or distance. Generally, in most countries and regions, numerous service providers offer RTCM protocol data, which are used to convey correction information. The RTK-GNSS combines RTK technology with standard GNSS systems, capitalizing on GNSS's global localisation abilities and RTK's high precision. It offers localisation services on a global scale.

Robots can obtain positioning information from IMU, wheel odometry, and RTK-GNSS, but each sensor has advantages and disadvantages. Multi-sensor fusion algorithms are needed to integrate different sensors and enhance localisation abilities to improve the robustness of robot localisation.

2.3. Low-Cost RTK-GNSS System

This research investigated a low-cost robot localisation system in unstructured farming environments. The system requires a base station, depicted in Figure 4, which gathers and transmits satellite data to the rover via digital radio. Concurrently, while obtaining data from the base, the rover also directly captures satellite information. The rover uses relative localisation principles to conduct real-time differential analyses with data from the base

and its readings, pinpointing its precise three-dimensional position with centimetre-level accuracy. This approach, costing GBP 220, involves two units: one is a base, and another is a rover that operates independently of internet connectivity and external differential localisation services.

Table 2 shows the cost comparison with other RTK-GNSS systems [18]. The absolute position coordinates are obtained through differential algorithms. Figure 5 shows the horizontal position accuracy. The localisation accuracy is stable at around 20 mm. Three sites were tested, and in Figure 6, the green lines show the trajectory of mapping the localisation data from the three test sites onto Bing's static satellite cloud map.

Table 2. Cost comparison with other RTK-GNSS systems [18].

Name	Cost
Piksi Multi Evaluation Kit	GBP 1785
Trimble R8s	GBP 7935
Spectra Precision SP80	GBP 97,211
Geomax Zenith35 Pro	GBP 77,726
Low-cost RTK-GNSS System	GBP 220

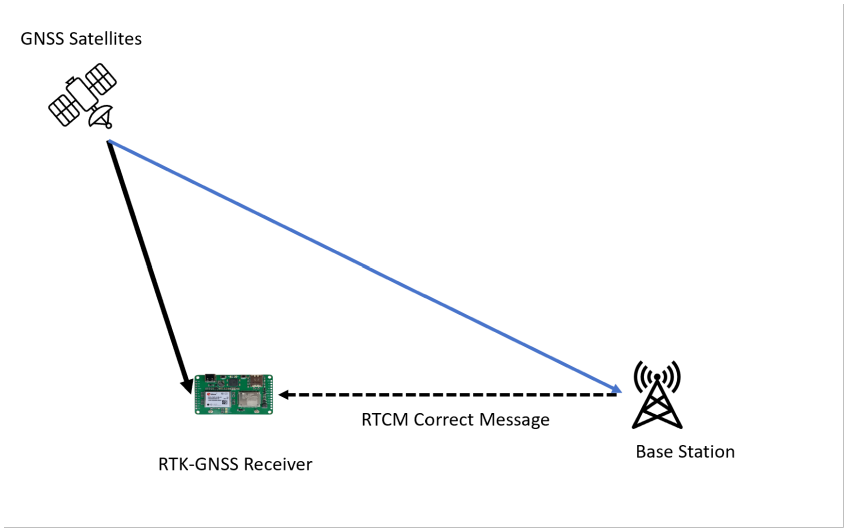

Figure 4. RTK-GNSS structure.

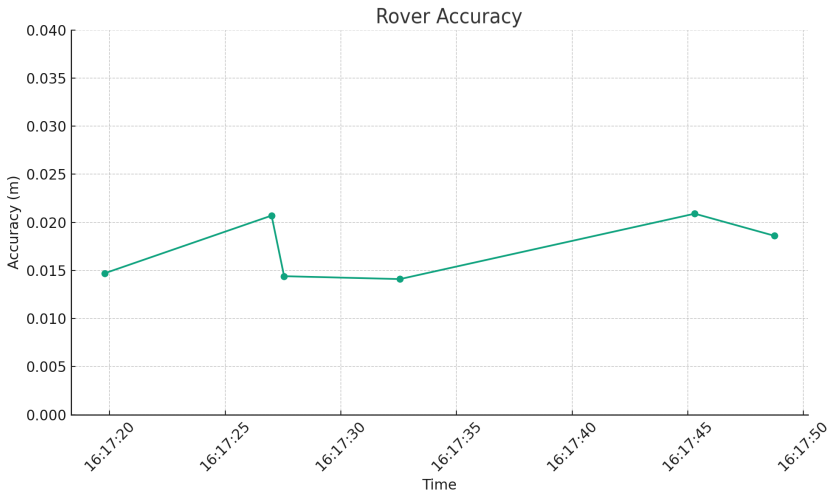

Figure 5. RTK-GNSS accuracy.

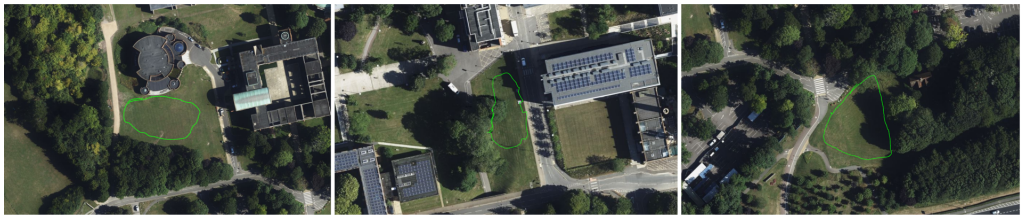

Figure 6. Test areas. The three images (**a**–**c**), correspond to the three test sites. (**a**) is surrounded by dense trees on two sides. It also includes several slopes and obstacles, simulating the dense vegetation in unstructured farm environments. (**b**) also includes dense vegetation on the left side and a steep slope area on the right. (**c**) is surrounded by high and dense trees, similar to the complex vegetative cover found in actual farms .

2.4. Robot Motion Model

To estimate the robot's position from the data from wheel odometry sensors, it is necessary to build a robot kinematic model. The data from wheel odometry are used as inputs, and the position and angle of the robot's motion can then be obtained through the kinematic model.

To simplify the kinematic model, three assumptions were made. Firstly, it was assumed that there is no slippage between the wheels and the ground, that the ground provides sufficient friction, and that the six wheels are parallel to each other. The kinematic model of a six-wheel differential drive robot is shown in Figure 7. Ideally, their speeds are synchronised for the wheels on the same side of the chassis. The chassis can perform circular motion around the instantaneous center of rotation (ICR). For the six wheels, the angular velocity of the circular motion is consistent, and the centre of this circular motion, the ICR, is always located on the y-axis extension of the chassis's center of gravity (COG). This is because the angular velocity ω_c directly influences the rate at which the chassis rotates around the ICR, thereby affecting the position of the ICR relative to the COG. The distance (DC) between the ICR (instantaneous center of rotation) and the COG (center of gravity) is constrained due to the differential drive mechanism, which dictates that changes in the velocity of the wheels alter the chassis's turning radius and thus shift the ICR's position along the y-axis. This constraint is related to the angular velocity ω_c of the circular motion, reflecting the dependency of the ICR's location on the rotational speed of the chassis, which in turn dictates the distance d_c between the ICR and the COG.

The velocity of the chassis is located at the center of mass (COM), represented by V_c. It is composed of components V_{cx} and V_{cy}. The velocities of the six wheels are denoted as v_1, v_2, v_3, v_4, v_5, and v_6 coming from target velocities v_{ix} and v_{iy} (where i = 1, 2, 3, 4, 5, 6). The distance between the left and right wheels is denoted as c. The formula for the angular velocity of circular motion is shown in Equation (1):

$$\omega_c = \frac{v_c}{d_c}, \tag{1}$$

where ω_c is the angular velocity of the circular motion, v_c is the linear velocity, and d_c is the radius of the circular motion. Set the angle between d_c and set the y-axis as α_c. From the perpendicular relationship between V_c and $ICR - COM$, it can be derived that $v_c \cos(\alpha_c) = v_{cx}$ and $v_c \sin(\alpha_c) = v_{cy}$. Summarising the above, the following constraint relationship is established:

$$\omega_c = \frac{v_c}{d_c} = \frac{v_c \cos(\alpha_c)}{d_c \cos(\alpha_c)} = \frac{v_{cx}}{d_{cy}}, \tag{2}$$

$$\omega_c = \frac{v_c}{d_c} = \frac{v_c \sin(\alpha_c)}{d_c \sin(\alpha_c)} = \frac{v_{cy}}{d_{cx}}. \tag{3}$$

Given the condition of consistent angular velocity among the six wheels of the rotating rigid body, Equations (2) and (3) can be generalised to Equations (4) and (5) as follows:

$$\omega_c = \frac{v_i}{d_i} = \frac{v_i \cos(\alpha_i)}{d_i \cos(\alpha_i)} = \frac{v_{ix}}{d_{iy}}, \tag{4}$$

$$\omega_c = \frac{v_i}{d_i} = \frac{v_i \sin(\alpha_i)}{d_i \sin(\alpha_i)} = \frac{v_{iy}}{d_{ix}}. \tag{5}$$

Equation (6) is obtained from Equations (2)–(5):

$$\omega_c = \frac{v_c}{d_c} = \frac{v_{cx}}{d_{cy}} = \frac{v_{cy}}{d_{cx}} = \frac{v_{ix}}{d_{iy}} = \frac{v_{iy}}{d_{ix}} \quad (i = 1, 2, 3, 4, 5, 6). \tag{6}$$

At the same time, d_i (where $i = 1, 2, 3, 4, 5, 6$) and d_c satisfy Equations (7) and (8) regarding their projection lengths on the x-axis and y-axis, respectively:

$$d_{1y} = d_{3y} = d_{cy} - \frac{c}{2}, \tag{7}$$

$$d_{4y} = d_{6y} = d_{cy} + \frac{c}{2}. \tag{8}$$

For a six-wheel differential drive chassis, the speeds of the left and right wheels are set as V_L and V_R, respectively. Under the condition that the speeds of the front and rear wheels are strictly synchronised, the constraint, as shown in Equations (9) and (10), can be established as follows:

$$VL = \omega_c \cdot (d_{cy} - \frac{c}{2}) = \omega_c d_{cy} - \omega_c \cdot \frac{c}{2} = v_{cx} - \omega_c \cdot \frac{c}{2}, \tag{9}$$

$$VR = \omega_c \cdot (d_{cy} + \frac{c}{2}) = \omega_c d_{cy} + \omega_c \cdot \frac{c}{2} = v_{cx} + \omega_c \cdot \frac{c}{2}, \tag{10}$$

$$\begin{bmatrix} v_x \\ \omega_c \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{c} & -\frac{1}{c} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix}. \tag{11}$$

Thus, the inverse kinematics formula can be obtained simply by performing an inverse transformation. The inverse kinematics model for a six-wheel differential drive system is shown in Equation (12):

$$\begin{bmatrix} V_L \\ V_R \end{bmatrix} = \begin{bmatrix} 1 & -\frac{c}{2} \\ 1 & \frac{c}{2} \end{bmatrix} \begin{bmatrix} V_{cx} \\ \omega_c \end{bmatrix}, \tag{12}$$

$$V_c = \frac{V_L + V_R}{2}. \tag{13}$$

The rotational speed of each wheel obtained through wheel speed encoders allows for the calculation of the robot's linear velocity V_c and angular velocity ω_c. Within a two-dimensional plane, V_c can be decomposed into components in the V_x and V_y directions:

$$\Delta x = \int V_c \cos(\theta)\, dt, \tag{14}$$

$$\Delta y = \int V_c \sin(\theta)\, dt. \tag{15}$$

Integrate the angular velocity ω_c to calculate the change in direction (yaw angle change):

$$\Delta\theta = \int \omega_c\, dt. \tag{16}$$

The calculated displacements Δx and Δy are applied to the current position of the robot. In contrast, the orientation change $\Delta\theta$ is applied to the current orientation to obtain the new pose of the robot:

$$x_{\text{new}} = x_{\text{old}} + \Delta x, \tag{17}$$

$$y_{\text{new}} = y_{\text{old}} + \Delta y, \tag{18}$$

$$\theta_{\text{new}} = \theta_{\text{old}} + \Delta\theta. \tag{19}$$

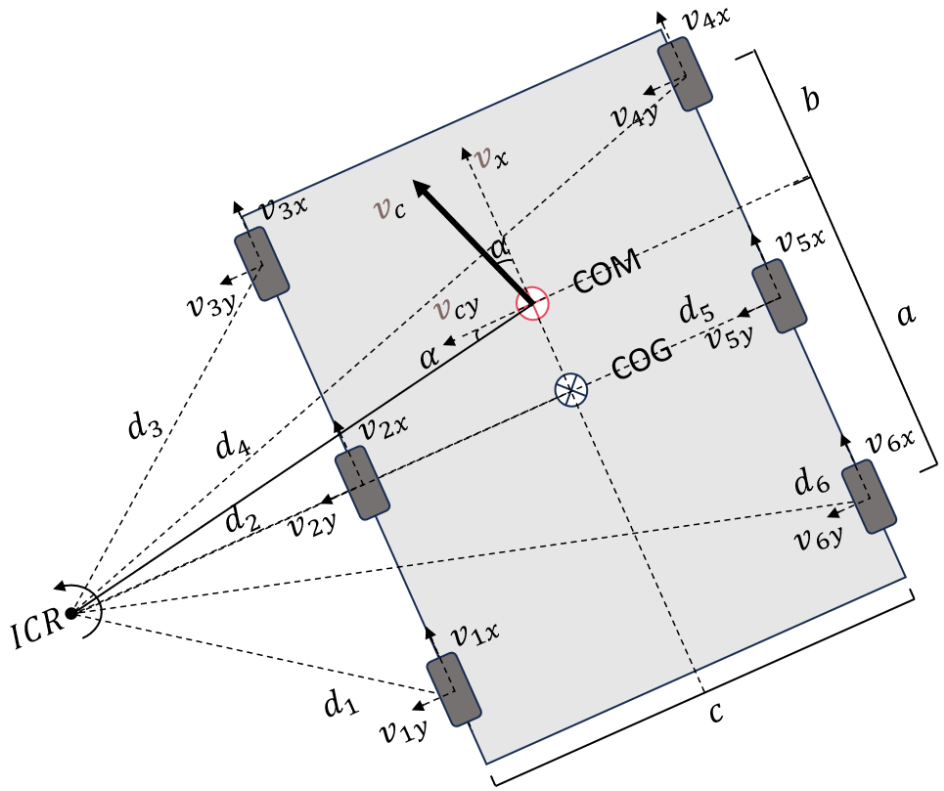

Figure 7. Robot kinematics model.

Wheel odometry can be used to obtain the wheel speeds on the robot's left and right sides. By substituting these into Equation (12), the robot's velocity can be calculated. Then, by inserting Equations (14)–(16) into Equations (17)–(19), the position and angle of the robot's movement can be determined, constituting the robot's pose.

2.5. Method 1: RTK-GNSS IMU Odometry Fusion with ESKF

IMU, wheel odometry, and GNSS have drawbacks in unstructured farm environments. The IMU can provide rapid updates of motion status, but it may drift over prolonged use. Wheel odometry can provide continuous motion information, but differential wheels may experience significant slippage and accumulate errors. GNSS signals can be obstructed, leading to significant errors or unavailability in certain areas. To improve the accuracy of robot localisation, multi-sensor fusion algorithms are needed to integrate data from different sensors and enhance localisation abilities.

The Kalman filter is widely used to estimate system states by combining a predictive model with observational data from multiple sensors [19]. However, compared to the extended Kalman filter (EKF), the error state Kalman filter (ESKF) offers more advantages in multi-sensor robot localisation. By linearising the error states, the ESKF provides higher precision, better robustness against sensor noise, and improved computational efficiency [20]. Therefore, this study is the first to use the ESKF to fuse data from low-cost IMU, wheel odometry, and low-cost RTK-GNSS sensors to enhance localisation accuracy in unstructured farm environments.

2.5.1. State Definition

To fuse the data from IMU, RTK-GNSS, and wheel odometry using the ESKF, the state equations and observation equations need to be established, the state vector and error state vector need to be defined, and the state covariance matrix and observation noise covariance matrix need to be initialised.

In the ESKF, the state vector X typically includes position, velocity, attitude (quaternion), accelerometer bias, gyroscope bias, and odometer bias. The state vector is defined as follows:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{q} \\ \mathbf{b}_a \\ \mathbf{b}_\omega \\ \mathbf{b}_{odom} \end{bmatrix}, \tag{20}$$

where p is the position vector (x, y, z), v is the velocity vector (v_x, v_y, v_z), q is the quaternion representing the attitude (q_w, q_x, q_y, q_z), b_a is the accelerometer bias, b_ω is the gyro bias, and b_{odom} is the odometer bias.

2.5.2. Continuous-Time State Equations

In the ESKF, state equations are used to describe the system state over time. The continuous-time state equations are:

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{21}$$

$$\dot{\mathbf{v}} = \mathbf{R}(\mathbf{q})(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + \mathbf{g}, \tag{22}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes (\omega_m - \mathbf{b}_\omega - \mathbf{n}_\omega), \tag{23}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{b_a}, \tag{24}$$

$$\dot{\mathbf{b}}_\omega = \mathbf{n}_{b_\omega}, \tag{25}$$

$$\dot{\mathbf{b}}_{odom} = \mathbf{n}_{b_{odom}}. \tag{26}$$

Here, $\mathbf{R}(\mathbf{q})$ is a rotation matrix represented by the quaternion $\mathbf{q}$, $\mathbf{a}_m$ is the acceleration measured by IMU, ω_m is the angular velocity measured by IMU, $\mathbf{g}$ is the acceleration of gravity, and $\mathbf{n}_a$, $\mathbf{n}_\omega$, $\mathbf{n}_{b_a}$, $\mathbf{n}_{b_\omega}$, and $\mathbf{n}_{b_{odom}}$ represent process noise.

2.5.3. Discrete State Equations

Since both the sensor data and the computer processing are performed at discrete time steps, the continuous-time state equations need to be discretised.

In classical physics, the relationship between position $\mathbf{p}$ and velocity $\mathbf{v}$ can be expressed by the following differential equation:

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{27}$$

$$\dot{\mathbf{v}} = \mathbf{a}, \tag{28}$$

where $\mathbf{a}$ is the acceleration. When discretizing, these continuous-time differential equations need to be converted into discrete-time difference equations.

Assume that the acceleration $\mathbf{a}$ is constant over a time step Δt. According to Newton's laws of motion, the change in velocity $\mathbf{v}$ can be expressed as:

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{a}_k \Delta t. \tag{29}$$

The change in position $\mathbf{p}$ can be obtained by integrating over the velocity. Over a time step Δt, assuming an initial velocity of $\mathbf{v}_k$ and an end velocity of $\mathbf{v}_{k+1}$, the change in position can be expressed as an integral over the velocity:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \int_{t_k}^{t_{k+1}} \mathbf{v}(t)\, dt. \tag{30}$$

Since the velocity $\mathbf{v}$ varies linearly, it can be approximated by trapezoidal integration:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \left(\mathbf{v}_k + \frac{1}{2}\mathbf{a}_k \Delta t\right)\Delta t. \tag{31}$$

Expanding this further, we obtain:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_k \Delta t + \frac{1}{2}\mathbf{a}_k \Delta t^2. \tag{32}$$

The acceleration $\mathbf{a}_k$ is obtained from the acceleration data measured by the IMU $\mathbf{a}_m$ minus the accelerometer bias $\mathbf{b}_a$, and it also requires consideration of the rotation matrix $\mathbf{R}(\mathbf{q})$ to transform the acceleration from the sensor coordinate system to the world coordinate system. Therefore, the position update equation and velocity update equation are:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_k \Delta t + 0.5\mathbf{R}(\mathbf{q}_k)(\mathbf{a}_m - \mathbf{b}_a)\Delta t^2, \tag{33}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{R}(\mathbf{q}_k)(\mathbf{a}_m - \mathbf{b}_a)\Delta t + \mathbf{g}\Delta t. \tag{34}$$

In the ESKF, attitude is usually represented by quaternions. Quaternions are a mathematical tool that avoids singularities and simplifies rotation calculations. It needs to derive the discrete-time update equation for quaternions to use quaternions for attitude updates.

1. Quaternion Representation of Attitude

A quaternion $\mathbf{q}$ represents a rotation and is defined as:

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix}, \tag{35}$$

where q_w is the real part and (q_x, q_y, q_z) are the imaginary parts.

2. Quaternion Differential Equation

The change in attitude is determined by angular velocity. Assuming the angular velocity ω is measured by the IMU, denoted as ω_m, and considering sensor bias and noise, the actual angular velocity is $\omega_m - b_\omega - n_\omega$.

The differential equation for the quaternion is:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix}, \tag{36}$$

where $\otimes$ denotes quaternion multiplication.

3. Discretizing the Quaternion Differential Equation

To discretise the quaternion differential equation, consider the change over a time step Δt. Assuming the angular velocity is constant over this time step, we can represent the change in the quaternion as the product of the initial quaternion and a small rotation quaternion.

3.1 Small Rotation Quaternion

The angular velocity $\boldsymbol{\omega} = \boldsymbol{\omega}_m - \mathbf{b}_\omega - \mathbf{n}_\omega$ over a small time step Δt can be approximated as:

$$\boldsymbol{\omega}\Delta t \approx \begin{bmatrix} 0 \\ \omega_x \Delta t \\ \omega_y \Delta t \\ \omega_z \Delta t \end{bmatrix}. \tag{37}$$

The corresponding small rotation quaternion is:

$$\Delta \mathbf{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\omega_x \Delta t \\ \frac{1}{2}\omega_y \Delta t \\ \frac{1}{2}\omega_z \Delta t \end{bmatrix}. \tag{38}$$

3.2 Quaternion Update

The new attitude quaternion $\mathbf{q}_{k+1}$ can be represented as:

$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \Delta \mathbf{q}, \tag{39}$$

where $\Delta \mathbf{q}$ is the small rotation quaternion over the time step Δt.

4. Using Exponential Mapping to Represent Small Rotations

To more accurately represent small rotations, an exponential mapping can be used:

$$\Delta \mathbf{q} = \exp\left(\frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{\omega}\Delta t \end{bmatrix}\right), \tag{40}$$

where $\exp(\cdot)$ is the exponential map for quaternions, which can be calculated using Rodrigues' rotation formula or the quaternion exponential formula.

5. Discrete-Time Attitude Update Equation

Combining the above derivations, the discrete-time attitude update equation is:

$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \exp\left(\frac{\boldsymbol{\omega}_m - \mathbf{b}_\omega}{2}\Delta t\right). \tag{41}$$

Accelerometer, gyroscope, and odometer deviations are assumed to be constant:

$$\mathbf{b}_{a_{k+1}} = \mathbf{b}_{a_k}, \tag{42}$$

$$\mathbf{b}_{\omega_{k+1}} = \mathbf{b}_{\omega_k}, \tag{43}$$

$$\mathbf{b}_{odom_{k+1}} = \mathbf{b}_{odom_k}. \tag{44}$$

2.5.4. Error State Equations

To use ESKF, the error state equations need to be defined, capturing the difference between the true state and the estimated state. The error state vector is defined as:

$$\delta \mathbf{x} = \begin{bmatrix} \delta \mathbf{p} \\ \delta \mathbf{v} \\ \delta \theta \\ \delta \mathbf{b}_a \\ \delta \mathbf{b}_\omega \\ \delta \mathbf{b}_{odom} \end{bmatrix}. \tag{45}$$

The error state equations are:

$$\delta\dot{\mathbf{p}} = \delta\mathbf{v}, \tag{46}$$

$$\delta\dot{\mathbf{v}} = -\mathbf{R}(\hat{\mathbf{q}})[\mathbf{a}_m - \hat{\mathbf{b}}_a]_\times \delta\grave{} - \mathbf{R}(\hat{\mathbf{q}})\delta\mathbf{b}_a + \mathbf{n}_a, \tag{47}$$

$$\delta\dot{\theta} = \omega_m - \hat{\mathbf{b}}_\omega + \mathbf{n}_\omega, \tag{48}$$

$$\delta\dot{\mathbf{b}}_a = \mathbf{n}_{ba}, \tag{49}$$

$$\delta\dot{\mathbf{b}}_\omega = \mathbf{n}_{b\omega}, \tag{50}$$

$$\delta\dot{\mathbf{b}}_{odom} = \mathbf{n}_{b_{odom}}, \tag{51}$$

where $[\mathbf{a}_m - \hat{\mathbf{b}}_a]_\times$ is the skew-symmetric matrix of the vector $\mathbf{a}_m - \hat{\mathbf{b}}_a$.

Discretizing the error state equations, we obtain:

$$\delta\mathbf{p}_{k+1} = \delta\mathbf{p}_k + \delta\mathbf{v}_k\Delta t, \tag{52}$$

$$\delta\mathbf{v}_{k+1} = \delta\mathbf{v}_k + \left(-\mathbf{R}(\hat{\mathbf{q}}_k)[\mathbf{a}_m - \hat{\mathbf{b}}_a]_k \delta\grave{}_k - \mathbf{R}(\hat{\mathbf{q}}_k)\delta\mathbf{b}_a\right)\Delta t, \tag{53}$$

$$\delta\grave{}_{k+1} = \delta\grave{}_k + (\omega_m - \hat{\mathbf{b}}_\omega)_k\Delta t, \tag{54}$$

$$\delta\mathbf{b}_{a_{k+1}} = \delta\mathbf{b}_{a_k}, \tag{55}$$

$$\delta\mathbf{b}_{\omega_{k+1}} = \delta\mathbf{b}_{\omega_k}, \tag{56}$$

$$\delta\mathbf{b}_{odom_{k+1}} = \delta\mathbf{b}_{odom_k}. \tag{57}$$

2.5.5. Prediction of the Error Covariance

The prediction step of the error covariance matrix is given by:

$$\mathbf{P}_{k+1} = \mathbf{F}_k\mathbf{P}_k\mathbf{F}_k^T + \mathbf{Q}_k, \tag{58}$$

where $\mathbf{F}_k$ is the Jacobian matrix of the state transition model, and $\mathbf{Q}_k$ is the process noise covariance matrix.

2.5.6. Measurements

The GNSS provides position information for state correction as follows:

$$\mathbf{z}_{gnss} = \mathbf{H}_{gnss}\mathbf{x} + \mathbf{v}_{gnss}, \tag{59}$$

where H_{gnss} is the measurement matrix, and v_{gnss} is the measurement noise.

Wheel odometry provides velocity information for state correction as follows:

$$\mathbf{z}_{odom} = \mathbf{H}_{odom}\delta\mathbf{x} + \mathbf{v}_{odom}, \tag{60}$$

where H_{odom} is the measurement matrix, and v_{odom} is the measurement noise.

2.5.7. Kalman Gain and State Update

Kalman gain is computed as follows:

$$\mathbf{K}_{gnss} = \mathbf{P}_{k+1}\mathbf{H}_{gnss}^T(\mathbf{H}_{gnss}\mathbf{P}_{k+1}\mathbf{H}_{gnss}^T + \mathbf{R}_{gnss})^{-1}, \tag{61}$$

$$\mathbf{K}_{odom} = \mathbf{P}_{k+1}\mathbf{H}_{odom}^T(\mathbf{H}_{odom}\mathbf{P}_{k+1}\mathbf{H}_{odom}^T + \mathbf{R}_{odom})^{-1}. \tag{62}$$

State update equations are as follows:

$$\delta\mathbf{x}_{k+1} = \delta\mathbf{x}_{k+1} + \mathbf{K}_{gnss}(\mathbf{z}_{gnss} - \mathbf{H}_{gnss}\delta\mathbf{x} \times k + 1), \tag{63}$$

$$\delta\mathbf{x}_{k+1} = \delta\mathbf{x}_{k+1} + \mathbf{K}_{odom}(\mathbf{z}_{odom} - \mathbf{H}_{odom}\delta\mathbf{x}_{k+1}). \tag{64}$$

Error covariance update equations are as follows:

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{gnss}\mathbf{H}_{gnss})\mathbf{P}_{k+1}, \tag{65}$$

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{odom}\mathbf{H}_{odom})\mathbf{P}_{k+1}. \tag{66}$$

The pseudo-code for the fusion of Odom IMU and RTK-GNSS processes via ESKF is shown in Algorithm 1.

Algorithm 1 ESKF Integrating IMU, RTK-GNSS, and Wheel Odometry

Initialization:
Set initial state estimate $x_0 = [p, v, q, b_a, b_\omega, b_{odom}]^T$ and initial state covariance P_0 ▷ *Initial position p, velocity v, orientation q, accelerometer bias b_a, gyroscope bias b_ω, odometry bias b_{odom}*
Set process noise covariance Q and measurement noise covariances R_{gnss}, R_{odom} ▷ *Uncertainty in model Q and sensors R*
Define the state transition function $f()$ and observation models $h_{gnss}()$, $h_{odom}()$ ▷ *Functions for state prediction and measurements*
for each time step $k = 1, 2, \ldots$ **do**
    Acquire data from sensors:
    Obtain angular velocity ω_{IMU} and acceleration a_{IMU} from the IMU ▷ *IMU provides rotation rate ω_{IMU} and acceleration a_{IMU}*
    Obtain wheel speeds v_L, v_R from the odometer ▷ *Wheel encoders measure speeds v_L and v_R*
    Obtain global localisation coordinates pos_{GNSS} from RTK-GNSS ▷ *GNSS gives accurate global position pos_{GNSS}*
    Compute control input and observation based on sensor data:
    Calculate control input $u_k = [v_c, \omega_c]^T$ based on ω_{IMU}, v_L, v_R ▷ *Compute linear v_c and angular ω_c velocities*
    Observation z_{gnss} from RTK-GNSS and z_{odom} from odometry and IMU ▷ *Combine pose data from IMU and position from GNSS*
    Prediction step:
    Predict state $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1})$ ▷ *Predict next state using motion model $f()$*
    Calculate predicted covariance $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q$ ▷ *Estimate uncertainty in prediction*
    Update step:
    if GNSS data available **then**
        Calculate observation residual $\tilde{y}_{gnss} = z_{gnss} - h_{gnss}(\hat{x}_{k|k-1})$ ▷ *Difference between observed and expected GNSS measurements*
        Calculate residual covariance $S_{gnss} = H_{gnss} P_{k|k-1} H_{gnss}^T + R_{gnss}$ ▷ *Uncertainty in GNSS observation residual*
        Calculate Kalman gain $K_{gnss} = P_{k|k-1} H_{gnss}^T S_{gnss}^{-1}$ ▷ *Weighting of prediction vs. GNSS observation*
        Update state estimate $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{gnss}\tilde{y}_{gnss}$ ▷ *Correct the prediction with GNSS data*
        Update state covariance $P_{k|k} = (I - K_{gnss} H_{gnss}) P_{k|k-1}$ ▷ *Update the estimate uncertainty*
    else
        Calculate observation residual $\tilde{y}_{odom} = z_{odom} - h_{odom}(\hat{x}_{k|k-1})$ ▷ *Difference between observed and expected odometry measurements*
        Calculate residual covariance $S_{odom} = H_{odom} P_{k|k-1} H_{odom}^T + R_{odom}$ ▷ *Uncertainty in odometry observation residual*
        Calculate Kalman gain $K_{odom} = P_{k|k-1} H_{odom}^T S_{odom}^{-1}$ ▷ *Weighting of prediction vs. odometry observation*
        Update state estimate $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{odom}\tilde{y}_{odom}$ ▷ *Correct the prediction with odometry data*
        Update state covariance $P_{k|k} = (I - K_{odom} H_{odom}) P_{k|k-1}$ ▷ *Update the estimate uncertainty*

2.6. Method 2: LIO-SAM

This subsection describes another localisation method using LiDAR sensors called LIO-SAM (LiDAR Inertial Odometry via Smoothing and Mapping). This is a SLAM framework that combines LiDAR and inertial measurement unit (IMU) data and that, at its core, is an optimisation method based on the factor graph, which is a kind of bipartite graph used to model the impact of sensor measurements on robot constraints on positional and environmental landmarks. Figure 8 shows the system structure of the LIO-SAM, which contains four factors [13].

The first type is the IMU pre-integration factor (orange), based on the IMU measurements between two consecutive keyframes. The IMU can provide high-frequency information about the robot's acceleration and angular velocity. By pre-integrating these IMU data, a rough estimate of the robot's motion between the two keyframes can be obtained. This estimation helps bridge the motion between keyframes, providing continuous constraints for changes in the robot's pose.

The second type is the LiDAR odometry factor (green), obtained by frame-to-frame matching of the LiDAR point cloud data from each keyframe with the data from the previous n keyframes. Based on the geometric correspondences between point clouds, this matching can provide accurate information about the robot's position changes. The LiDAR odometry factors assist in fine-tuning the pose of the keyframes to ensure optimal alignment between LiDAR point clouds.

The third type is the GPS factor (yellow), derived from the GPS measurements for each keyframe, offering global localisation information. GPS data can provide a global reference for the robot, helping to correct accumulated drift errors and ensure the accuracy of long-term navigation. GPS factors are particularly useful in outdoor environments, especially in open spaces with good GPS signal reception.

The fourth type is the loop closure factor (black), a critical component of SLAM systems for identifying when the robot revisits previously explored areas. Upon detecting a potential loop closure, the system selects the candidate loop closure keyframe and its temporally adjacent $2m + 1$ keyframes for frame-to-frame matching. If the matching is successful, loop closure factors are added, connecting the current keyframe with the loop closure candidate keyframe. These factors help correct cumulative errors in the map, ensuring global consistency.

These four factors form the factor graph of the LIO-SAM. The LIO-SAM can fuse IMU, LiDAR, and GPS data by optimizing this factor graph, achieving high-precision real-time 3D mapping and localisation.

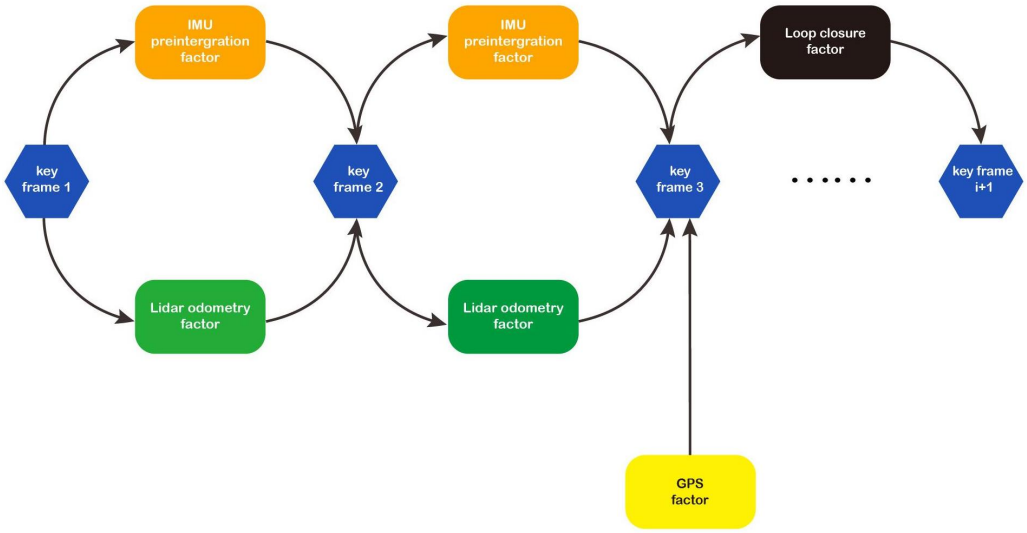

Figure 8. LIO-SAM factor graph structure [13].

3. Experiments

The experiments were conducted at three test sites near the University of Sussex campus with dense vegetation and multiple terrains, similar to unstructured farming environments. As shown in Figure 9, the University of Sussex is situated in South Downs National Park, with a wide range of grassland terrains.

Test Site A shown in Figure 10 is surrounded by dense trees on two sides. It also includes several slopes and obstacles, simulating the dense vegetation in unstructured farm environments.

Figure 9. Experiments at the University of Sussex, located in the South Downs National Park.

Figure 10. Test Site A.

Test Site B is shown in Figure 11. It also includes dense vegetation on the left side and a steep slope area on the right. The significant incline in this area is very similar to the undulating terrain found in farms and tested the robot's performance in unstructured terrain conditions.

Test Site C, shown in Figure 12, is surrounded by high and dense trees, similar to the complex vegetative cover found in actual farms.

A pre-planned trajectory was established for the robot to follow at each test site. To ensure consistency reproducibility and clarity of the experiments, the following steps were followed:

Figure 11. The slope in Test Site B.

Figure 12. The dense vegetation in Test Site C.

1. Initialisation: Before starting each experiment, all sensors were checked and calibrated to ensure they were functioning correctly.
2. Path Tracking: The robot moved along the pre-planned trajectory while continuously collecting sensor data.
3. Data Recording: All sensor data, including wheel odometry, IMU, LiDAR, and RTK-GNSS data, were recorded by the "rosbag record" command during the robot's movement; "rosbag" is a popular ROS tool for recording and broadcasting ROS messages. It allows the user to record data about the ROS topics being posted and store it in a .bag file.
4. Data Analysis: The collected data were analysed using the EVO tool to calculate the absolute pose error (APE) metrics for each method. To verify the robustness of the proposed algorithm when the GNSS signal is lost, the GNSS signal was manually switched off at a random interval to simulate the situation when the GNSS signal is lost. As shown in Figure 13, the green trajectories in the three test areas of the figure are the RTK-GNSS trajectories, and the trajectory breaks are where the RTK-GNSS topics were randomly turned off manually.
5. Comparison of Results: The localisation accuracies of the ESKF-based method and the LIO-SAM algorithm were compared across different test sites to evaluate their performance in various environments.

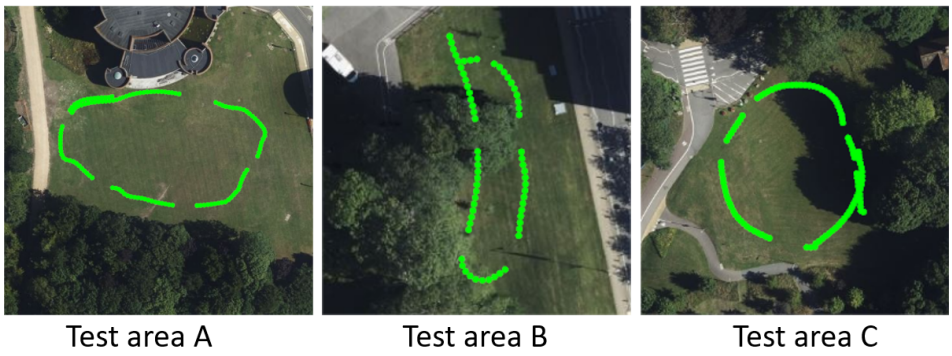

Figure 13. Trajectories with the RTK-GNSS topics turned off. The green lines in the test areas are the trajectories of the robot. The discontinuous parts of the trajectories are areas where GNSS was manually switched off.

3.1. Evaluation of Localisation Accuracy of Method 1 and Method 2

The effectiveness of the sensor fusion for the robot localisation method used in this paper, in comparison to LIO-SAM localisation, was verified in three areas labelled a, b, and c. These areas provided a range of conditions for a thorough evaluation. Figure 14 shows the experimental environments. The trajectory accuracy was evaluated using the EVO (Evaluation of Visual Odometry) tool, a standard tool in the fields of robotics and computer vision [21]. It processes various trajectory data formats and offers extensive data analysis and visualisation abilities. The evaluation criterion was APE (absolute pose error), which measures the error between the estimated and the reference trajectories. For this study, the trajectory from RTK-GNSS served as the reference, ensuring a reliable baseline for the accuracy assessment of the proposed localisation method.

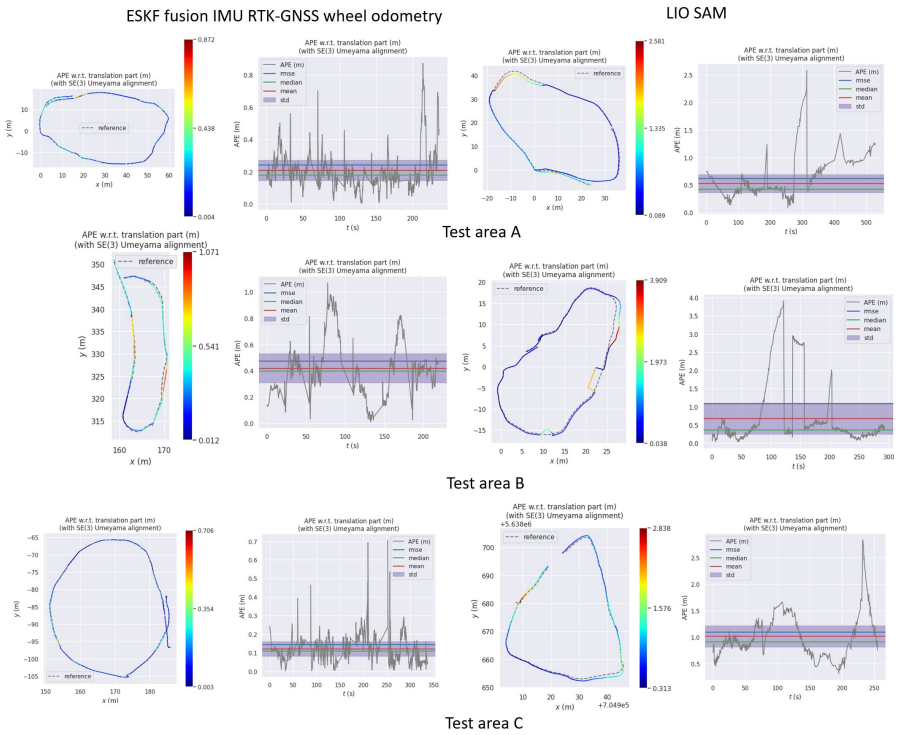

Figure 14. Absolute pose error (APE). Test A, Test B, and Test C are data obtained from three different fields. The two columns on the left are data obtained by fusing IMU, RTK-GNSS, and wheel odometry using the ESKF. The two columns on the right are obtained from the LIO-SAM algorithm. The red boxed parts indicate that a period is used to manually switch off the RTK-GNSS data while playing the "rosbag" data.

3.2. Real-Time Mapping

In this study, the LIO-SAM algorithm was deployed on the robotic platform. The imu-utils tool was employed to calibrate the IMU [22]. The calibration parameters are shown in Table 3. RoboSense RS-16, a low-cost LiDAR, was used in this research. However, the laser point cloud emitted by the RoboSense LiDAR has a different beam sequence and format than the Velodyne point cloud, which is the default in the algorithm. Therefore, the RS-to-Velodyne package was used to convert the point cloud data into the Velodyne data.

Experiments were conducted in three structured farm sites to validate the algorithm's effectiveness. Area A is a large area with irregular terrain, generally sloping from lower to higher elevations, surrounded by trees, and occasionally traversed by pedestrians. Area B has a steeper slope, showcasing the diversity of farm terrains. Area C is flat but characterised by dense vegetation. The mapping results are shown in Figure 15. For a more intuitive presentation of these complex agricultural environments, Figure 16 maps the point clouds onto Bing static maps.

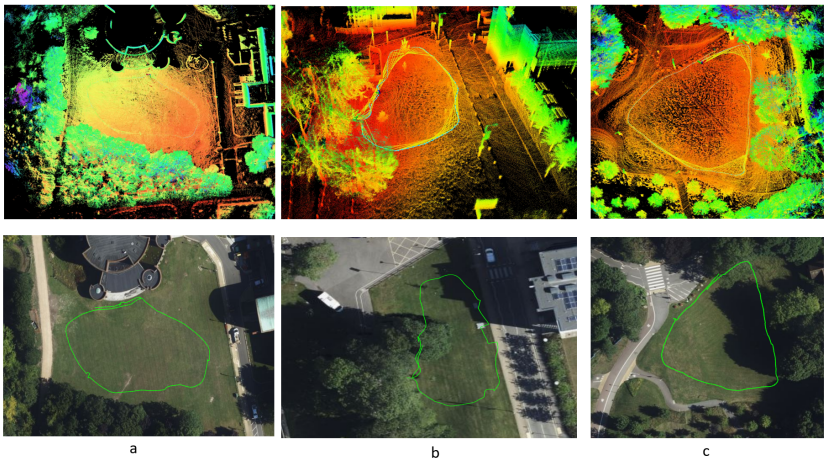

Figure 15. Point cloud and satellite cloud maps of different test areas obtained by the LIO-SAM algorithm. The three images (**a**–**c**), correspond to the three test sites. The top row shows the constructed 3D point cloud maps for each site, while the bottom row displays the projections of the robot's trajectory onto satellite maps during the tests.

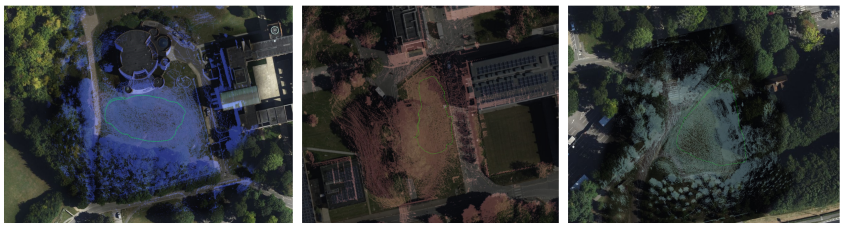

Figure 16. Point clouds mapped onto static maps.

Table 3. Parameters of the IMU sensor after calibration.

Measurement	Normal (n)	Random Walk (w)
Avg-axis	$9.1644814637628646 \times 10^{-4}$	$1.534994316938078 \times 10^{-5}$
x-axis	$6.2330144407217404 \times 10^{-4}$	$7.7852973316884239 \times 10^{-6}$
y-axis	$1.5193967801678358 \times 10^{-3}$	$2.3529647902414388 \times 10^{-5}$
z-axis	$6.0664621488884952 \times 10^{-4}$	$1.4734884274061423 \times 10^{-5}$
Avg-axis	$3.5460373043123113 \times 10^{-2}$	$3.035990452459453 \times 10^{-4}$
x-axis	$4.252986911819568 \times 10^{-2}$	$2.1006657093059140 \times 10^{-4}$
y-axis	$3.411867306809651 \times 10^{-2}$	$3.1010348121814275 \times 10^{-4}$
z-axis	$2.9739382704440123 \times 10^{-2}$	$3.9061766142504945 \times 10^{-4}$

4. Results and Discussion

This study investigated a localisation method that uses ESKF to integrate data from wheel odometry, a low-cost IMU, and a low-cost RTK-GNSS. The accuracy of the estimated trajectories is also compared with the LIO-SAM algorithm using the absolute pose error (APE) metric from the Evaluation of Visual Odometry (EVO) and SLAM tools. EVO is a tool for evaluating the performance of visual odometry and SLAM algorithms, providing a convenient way to compare and evaluate trajectories generated by different algorithms [21]. The results obtained from three different test areas are displayed in Figure 14, where the data from the ESKF-based method and the LIO-SAM algorithm are shown in the left and right columns, respectively. Specifically, the first and third columns present the reference trajectories compared to the estimated trajectories by the two methods, while the second and fourth columns show the corresponding trajectory errors.

As shown in Table 4, the first to third rows are the trajectory error evaluation parameters of the methods presented in this paper, and the fourth to sixth rows are the trajectory error evaluation parameters of the LIO-SAM method. Here, 'Max' refers to the maximum error, 'Mean' refers to the average error, 'Median' refers to the middle value in the list of all error values, 'Min' refers to the minimum error, 'RMSE' refers to the root mean square error, and 'Std' refers to the standard deviation.

The errors in Table 4 are obtained by taking the GNSS trajectory as the ground truth trajectory and comparing the estimated trajectories of the proposed two algorithms. Divide the RMSE value by the total length of the trajectory and multiply by 100% to obtain the error percentage. By calculating the percentage errors for different experimental sites, the ESKF method's percentage errors ranged from 0.11% to 0.54%, while the LIO-SAM method's percentage errors ranged from 0.29% to 0.67%.

Table 4. Comparison of APE parameters and costs of Method 1 and Method 2.

Method	Max /m	Mean /m	Median /m	Min /m	RMSE /m	Std /m	Percentage Error %	Sensors	Test Site	Cost
ESKF-based method	0.8724	0.2079	0.1790	0.0035	0.2430	0.1257	0.14%	RTK-GNSS	Site A	GBP 220
	1.0706	0.4197	0.3986	0.0116	0.4747	0.2218	0.54%	IMU	Site B	GBP 29
	0.7061	0.1218	0.1106	0.0025	0.1449	0.0785	0.11%	Odom	Site C	
LIO-SAM	2.5812	0.5296	0.4397	0.0805	0.6211	0.3245	0.29%	LiDAR	Site A	GBP 670
	3.9087	0.6763	0.3654	0.0375	1.09	0.8581	0.87%	IMU	Site B	GBP 29
	2.8383	1.0192	0.9165	0.3134	1.0971	0.4061	0.67%	-	Site C	

Absolute pose errors (APEs) obtained from three different test areas are displayed in Figure 14. The left and right columns present the data from the ESKF-based method and the LIO-SAM algorithm, respectively. The first and third columns show the reference trajectories compared to the estimated trajectories by the two methods, while the second and fourth columns display the corresponding trajectory errors. From Test Area A to Test Area C, it can be seen that the errors of the ESKF-based method are smaller than the LIO-SAM method.

Figure 16 shows that the LIO-SAM algorithm successfully generated high-quality point cloud maps, accurately reflecting the terrain undulations and environmental features. The algorithms exhibited robust performance even when GNSS signals were lost, without any drift in the point cloud maps, accurately reflecting the terrain and environmental conditions of the test sites.

The ESKF-based method achieved a high localisation accuracy, and it is more cost-effective and robust when the GNSS signal is lost. It also provides a viable alternative for low-cost agricultural robot localisation. The LIO-SAM algorithm, on the other hand, successfully generated high-quality point cloud maps, accurately reflecting the terrain and environmental features, even when GNSS signals were intermittent.

5. Conclusions

This study compared two localisation methods that use the ESKF to integrate data from wheel odometry, a low-IMU, and a low-cost RTK-GNSS, as well as the LIO-SAM algorithm using a low-cost IMU and RoboSense 16-channel LiDAR sensor. The performance of these methods was compared using the absolute pose error (APE) metric from the Evaluation of Visual Odometry (EVO) and SLAM tools. The experimental results indicated that the EKF-based localisation error is smaller than the LIO-SAM algorithm. However, the cost of the ESKF-based method is only 36% of the LIO-SAM system. In conclusion, the proposed ESKF-based localisation method offers a cost-effective and feasible solution for agricultural robot localisation in unstructured environments. It demonstrates the potential for applications in unstructured farm environments and enables the adoption of autonomous robots in the agriculture sector.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, writing—original draft preparation, writing—review and editing, visualisation, C.L. Review and supervision, B.K.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the University of Sussex.

Data Availability Statement: The dataset used in this study was collected using our robot platform. For researchers interested in implementation details or access to the processed data, the corresponding code and dataset are available upon request from the corresponding author.

Acknowledgments: We express our sincere appreciation to the University of Sussex and Zhejiang Gongshang University for their financial support. Our gratitude also extends to RTKFnet Company for providing NTRIP services for this research.

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

Abbreviations

The following abbreviations are used in this manuscript:

IMU	inertial measurement unit
RTK-GNSS	real-time kinematic global navigation satellite system
ESKF	error state Kalman filter
Odom	odometry
LIO-SAM	LiDAR-Inertial Odometry via Smoothing and Mapping

References

1. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
2. Pini, M.; Marucco, G.; Falco, G.; Nicola, M.; De Wilde, W. Experimental testbed and methodology for the assessment of RTK GNSS receivers used in precision agriculture. *IEEE Access* **2020**, *8*, 14690–14703. [CrossRef]
3. Dong, F.; Heinemann, W.; Kasper, R. Development of a row guidance system for an autonomous robot for white asparagus harvesting. *Comput. Electron. Agric.* **2011**, *79*, 216–225. [CrossRef]
4. Koo, G.; Kim, K.; Chung, J.Y.; Choi, J.; Kwon, N.Y.; Kang, D.Y.; Sohn, H. Development of a high precision displacement measurement system by fusing a low cost RTK-GPS sensor and a force feedback accelerometer for infrastructure monitoring. *Sensors* **2017**, *17*, 2745. [CrossRef] [PubMed]
5. Titterton, D.; Weston, J.L. *Strapdown Inertial Navigation Technology*; IET: London, UK, 2004; Volume 17.
6. Aggarwal, P. *MEMS-Based Integrated Navigation*; Artech House: Boston, MA, USA, 2010.
7. Angrisano, A. GNSS/INS Integration Methods. Ph.D. Thesis, Universita'degli Studi di Napoli PARTHENOPE, Naple, Italy, 2010.
8. Falco, G.; Nicola, M.; Pini, M. Positioning based on tightly coupled multiple sensors: A practical implementation and experimental assessment. *IEEE Access* **2018**, *6*, 13101–13116. [CrossRef]
9. Falco, G.; Pini, M.; Marucco, G. Loose and tight GNSS/INS integrations: Comparison of performance assessed in real urban scenarios. *Sensors* **2017**, *17*, 255. [CrossRef] [PubMed]
10. Bevly, D.M.; Cobb, S. *GNSS for Vehicle Control*; Artech House: Boston, MA, USA, 2010.
11. Lin, J.; Peng, J.; Hu, Z.; Xie, X.; Peng, R.; et al. Orb-slam, imu and wheel odometry fusion for indoor mobile robot localization and navigation. *Acad. J. Comput. Inf. Sci* **2020**, *3*, 030114.

12. Zhang, C.; Zhan, Q.; Wang, Q.; Wu, H.; He, T.; An, Y. Autonomous dam surveillance robot system based on multi-sensor fusion. *Sensors* **2020**, *20*, 1097. [CrossRef] [PubMed]
13. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2021; pp. 5135–5142.
14. Bula, J.; Derron, M.H.; Mariethoz, G. Dense point cloud acquisition with a low-cost Velodyne VLP-16. *Geosci. Instrum. Methods Data Syst.* **2020**, *9*, 385–396. [CrossRef]
15. Kaplan, E.D.; Hegarty, C. *Understanding GPS/GNSS: Principles and Applications*; Artech House: Boston, MA, USA, 2017.
16. Baybura, T.; Tiryakioğlu, İ.; Uğur, M.A.; Solak, H.İ.; Şafak, Ş. Examining the accuracy of network RTK and long base RTK methods with repetitive measurements. *J. Sens.* **2019**, *2019*, 1–12. [CrossRef]
17. Inal, C.; Bulbul, S.; Bilgen, B. Statistical analysis of accuracy and precision of GNSS receivers used in network RTK. *Arab. J. Geosci.* **2018**, *11*, 227. [CrossRef]
18. Moeller, R.; Deemyad, T.; Sebastian, A. Autonomous navigation of an agricultural robot using RTK GPS and Pixhawk. In Proceedings of the 2020 Intermountain Engineering, Technology and Computing (IETC), Orem, UT, USA, 2–3 October 2020; pp. 1–6.
19. Konatowski, S.; Pieni, A.T. A comparison of estimation accuracy by the use of KF, EKF & UKF filters. *WIT Trans. Model. Simul.* **2007**, *46*, 11.
20. Li, Z.; Zhang, Y. Constrained ESKF for UAV positioning in indoor corridor environment based on IMU and WiFi. *Sensors* **2022**, *22*, 391. [CrossRef] [PubMed]
21. Grupp, M. evo: Python Package for the Evaluation of Odometry and SLAM. 2017. Available online: https://github.com/MichaelGrupp/evo (accessed on 10 April 2024).
22. Woodman. An Introduction to Inertial Navigation (No. UCAM-CL-TR-696). 2007. Available online: https://github.com/gaowenliang/imu_utils (accessed on 27 December 2023).