UAV Localization Algorithm Based on Factor Graph Optimization in Complex Scenes

Jun Dai 1,2, Songlin Liu 1, Xiangyang Hao 1,*, Zongbin Ren 1 and Xiao Yang 3

1 Institute of Geospatial Information, Information Engineering University, Zhengzhou 450001, China
2 School of Aerospace Engineering, Zhengzhou University of Aeronautics, Zhengzhou 450001, China
3 Dengzhou Water Conservancy Bureau, Dengzhou 474150, China
* Correspondence: xiangyanghao2004@163.com; Tel.: +86-135-9804-9970

Abstract: With the increasingly widespread application of UAV intelligence, the need for autonomous navigation and positioning is becoming more and more important. To solve the problem that UAV cannot perform localization in complex scenes, a new multi-source fusion framework factor graph optimization algorithm is used for UAV localization state estimation in this paper, which is based on IMU/GNSS/VO multi-source sensors. Based on the factor graph model and the iSAM incremental inference algorithm, a multi-source fusion model of IMU/GNSS/VO is established, including the IMU pre-integration factor, IMU bias factor, GNSS factor, and VO factor. Mathematical simulations and validations on the EuRoC dataset show that, when the selected sliding window size is 30, the factor graph optimization (FGO) algorithm can not only meet the requirements of real time and accuracy at the same time, but it also achieves a plug-and-play function in the event of local sensor failures. Finally, compared with the traditional federated Kalman algorithm and the adaptive federated Kalman algorithm, the positioning accuracy of the FGO algorithm in this paper is improved by 1.5–2-fold, and can effectively improve autonomous navigation system robustness and flexibility in complex scenarios. Moreover, the multi-source fusion framework in this paper is a general algorithm framework that can satisfy other scenarios and other types of sensor combinations.

Keywords: UAV; multi-source fusion; factor graph optimization; robustness

1. Introduction

With the increasing demand for unmanned, intelligent, and autonomous features in various fields, UAVs (unmanned aerial vehicles) perform well in high-risk, complex, and repetitive tasks, so this has become a research hotspot and has developed rapidly. Whether in the military field or in civilian applications, UAV technology has higher and higher requirements for the accuracy of the navigation system [1,2]. A single navigation method cannot meet the requirements of UAV navigation accuracy and robustness. In complex scenes, combined and multi-source fusion navigation methods can provide more accurate and robust navigation and positioning results [3,4].

Multi-source sensor information fusion algorithms include the weighted average method, maximum likelihood estimation, least squares, Kalman filtering, D-S evidence theory, FGO, etc. [4]. In a complex environment, because some sensors are prone to failure, the system is required to realize asynchronous heterogeneous navigation source data fusion with the plug-and-play function. Most of the traditional multi-sensor information fusion methods use the integrated navigation method based on a federated Kalman filter (FKF). Although this method can fuse sensor information of different rates and calculate the navigation solution in real time through the data synchronization processing method, it is often necessary to discard a part of the measured values in order to maintain synchronization of the data, which will result in a waste of information. At the same time, the standard Kalman filter can only solve linear problems, and most sensor models contain nonlinear components [5,6].
The multi-sensor information fusion navigation algorithm based on a factor graph can solve the problems of traditional methods and has the advantage of plug and play. The fusion framework based on the factor graph model can effectively solve the asynchronous problem in data fusion, which has good scalability for multi-sensors and can be flexibly configured for sensors [7–10].

In 2007 and 2010, Dr. Levinson [11,12] of Stanford University proposed a method based on factor graph optimization to achieve map-based high-precision positioning. Ding et al. [13] made a major change to the data fusion framework, from filtering to factor graph optimization. Pfeifer et al. [14] uses a Gaussian mixture model to model the GNSS error model, and the final positioning accuracy is significantly improved.

Wang et al. [15] conducted research on the key technologies of all-source navigation and constructed a multi-sensor fusion framework based on factor graphs. The sensors involved include IMU, GPS, barometric altimeter, and optical flow sensors. Aiming at the collaborative navigation problem of densely clustered UAVs, Chen et al. [16] proposed a method for the collaborative navigation of UAV swarms based on factor graph optimization. The simulation test results show that the proposed method can effectively improve the positioning accuracy in multi-UAV application scenarios such as dense swarms. Tang et al. [17] defined the IMU, BDS, and odometer measurement factors, and constructed a multi-sensor fusion framework based on factor graphs. Gao et al. [18] constructed a factor graph model of the INS/GNSS/OD integrated navigation system, which can continuously and stably output high-precision navigation results, meeting the requirements of the vehicle-mounted INS/GNSS/OD system. Indelman et al. [19] demonstrated the proposed method in a simulated environment using IMU, GPS, and stereo vision measurements, and compared it to the optimal solution obtained by a full non-linear batch optimization, and to a conventional extended Kalman filter (EKF).

The above literature all focus on the establishment of the basic model and framework of the factor graphs. In order to study the asynchronous fusion problem of multi-source sensors, Xu et al. [20] proposed a multi-sensor information fusion method based on a factor graph, to fuse all available asynchronous sensor information, and to efficiently and accurately calculate a navigation solution. Considering the robustness of complex scenarios, Wei et al. [21] constructed an INS/GPS/OD factor graph model using factor graph technology, designed a dynamic weight function, and adjusted the weight of each factor reasonably and dynamically, thereby improving the navigational performance and robustness of the factor graph algorithm.

Artificial general intelligence (AGI) is a critical method to solve the problem that UAV cannot perform localization in complex scenes. With the increasingly widespread application of UAV intelligence, the need for autonomous navigation and positioning is becoming more and more important. AGI is a complementary approach for the factor graph optimization algorithm. Spiking Neural Networks (SNNs) are a way to solve problems such as high energy consumption in current machine learning techniques. Yang et al. [22] proposed a new spiking-based framework with minimum error entropy called MeMEE, which uses entropy theory. A gradient-based online meta-learning scheme is constructed in a recurrent SNN architecture to improve the applicability of a spike-based online meta-learning model for robust learning based on spatiotemporal dynamics and excellent machine learning theory. Aiming at the problem that there is a large gap between the learning performance of SNN and artificial neural network when there are few samples, Yang et al. [23] proposed Heterogeneous Ensemble-based Pulse-Driven Few-Shot Online Learning (HESFOL), which uses entropy theory to build a recurrent SNN architecture based on the gradient few-shot learning scheme, thereby improving the ability of SNN few-shot learning. In the field of text recognition, textual information has been shown to play an active role in recommender systems. Liu et al. [24] propose an Adaptive Attention Module (SAM), which reduces the potential selection bias caused by textual information by adjusting selection bias by capturing contextual information according to its representation. Aiming at the credit assignment problem of adjusting the weights of each neuron based on network routing
error information in neuromorphic computing, Yang et al. [25] proposed a novel dendritic event-based processing (DEP) algorithm, which uses two dendrites with partially separated dendrites. Chamber leakage integrates and fires neurons, effectively solving the credit assignment problem.

It can be seen from the above, that compared with the filtering algorithm, factor graph optimization inherits the idea of iteratively finding the optimal solution in graph optimization and has a higher precision. Compared with the traditional optimization algorithm, the factor graph optimization adopts the incremental reasoning algorithm, which has strong real-time performance [26,27]. The FGO can better solve the nonlinear problem of some state equations and observation equations in the navigation system, which lays the foundation for the realization of high-precision and robust positioning and navigation technology [28–31].

However, most of the previous studies are the comparison of the factor graph optimization algorithm and FKF, and a lack of comparison with other improved FKF algorithms. Moreover, most of these studies focus on factor graph optimization algorithms for state estimation, which do not consider the effect of window size on accuracy and real-time performance. In this paper, we start from the general adaptability of the algorithm, and use the data set collected in actual experiments to verify the algorithm. Further considering the adjustment of the sliding window, the optimal window size is chosen to balance accuracy and time. Specifically, we make the following contributions.

We propose a method for navigating and localizing UAVs using factor graphs for state estimation. By analyzing the IMU pre-integration factor, IMU (inertial measurement unit) bias factor, GNSS (Global Navigation Satellite Systems) factor, and the VO (visual odometry—where the position and attitude data are obtained by solving camera image poses) factor model—the IMU/GNSS/VO factor graph framework is constructed.

We perform two types of experiments to verify the effectiveness of the proposed factor graph framework in different scenarios, including mathematical simulation experiments and EuRoC datasets.

We balance time and accuracy by setting the size of the sliding window. By comparing the state estimation results of the factor graph, the federated Kalman filter, and the adaptive Kalman filter, the robustness of the factor graph algorithm in this paper is verified.

The rest of this paper is organized as follows: The factor graph model and the iSAM incremental inference algorithm are introduced in Section 2. In Section 3, a factor graph multi-source fusion model framework based on IMU/GNSS/VO is established. In Section 4, the traditional federated Kalman filter and the adaptive federated Kalman are listed for the comparative analysis of subsequent mathematical simulation experiments. In Section 5, through mathematical simulation experiments, the accuracy performances of the three algorithms are discussed and analyzed on the basis of the selected sliding window size. In Section 6, the effectiveness of the FGO algorithm is further demonstrated through dataset validation. Finally, conclusions and further research arrangements are drawn in Section 7.

2. Factor Graph Model

2.1. Factor Graph Algorithm

The factor graph model [7,32–34] is expressed as (1).

\[ G = (F, X, E) \]  

where \( X \) represents a set of variable nodes, \( F \) represents a set of factor nodes, and \( E \) represents a set of all edges connecting nodes.

The purpose of the factor graph optimization algorithm is to find the maximum posterior estimated probability.

\[ \hat{X} = \arg \max_{X} \prod_{k, i} P(Z_{k, i} | X_{k}) \prod_{k} P(X_{k} | X_{k-1}, u_{k}) \]
Where \( Z_{k,i} \) represents the observed value, \( \hat{X} \) represents the maximum posterior estimate, \( P(Z_{k,i}|X_k) \) represents the observed probability density, and \( P(X_k|X_{k-1}, u_k) \) represents the prior probability density.

By equivalent conversion to the solution factor sum product, a factor graph is thus defined as a decomposition of the function \( f_i(X_i) \) of the population of variables. The following Equation (3) is obtained.

\[
f(X) = \prod_i f_i(X_i)
\]  

(3)

The factor graph algorithm is the optimal estimated value of the solution, so that the function takes the maximum value.

\[
\hat{X} = \arg \max_X (\prod_i f_i(X_i))
\]  

(4)

The error model cost function is defined as (5).

\[
\|e\|^2 \triangleq \exp(-\frac{1}{2} \|h_i(X_i) - z_i\|^2_{\Sigma_i})
\]  

(5)

where \( h_i(X_i) \) is the observation function. Assuming that the above noise is a Gaussian noise model, \( \|e\|_2 = e^T \Sigma^{-1} e \) represents the Mahalanobis distance, \( \Sigma \) represents the covariance matrix, and (5) is transformed into (6) as follows.

\[
f_i(X_i) \propto \exp(-\frac{1}{2} \|h_i(X_i) - z_i\|^2_{\Sigma_i})
\]  

(6)

To sum up, (4) is transformed into a standard nonlinear least squares problem such as (7), whereby the optimal state value of the factor graph is obtained by optimizing the minimized error function, as follows.

\[
\hat{X} = \arg \min_X (\Sigma_i \|h_i(X_i) - z_i\|^2_{\Sigma_i})
\]  

(7)

2.2. Incremental Smoothing and Mapping (iSAM)

The factor graph model is a dynamic graph model that increases with time. Considering the real-time requirements, the method of iSAM should be adopted.

Equation (7) is expanded by Taylor to obtain a linear least squares problem, and the state update variables are as follows.

\[
d^* = \arg \min_{\delta} \left\| H_i \delta_i - \left\{ z_i - h_i(X_i^0) \right\} \right\|^2_{\Sigma_i}
\]  

(8)

where \( \delta_i = X_i - X_i^0 \) represents the state update vector, \( H_i = \frac{\partial h_i(X_i)}{\partial X_i} \bigg|_{X_i^0} \) represents the Jacobian measurement, \( d^* \) represents a local linear solution, and \( z_i - h_i(X_i^0) \) is the difference between the actual and predicted observations.

After (8) is whitened, it is finally transformed into a standard least squares optimization solution, as shown in (9).

\[
d^* = \arg \min_{\delta} \left\| A_i \delta_i - b_i \right\|^2_2 = \arg \min_{\delta} \left\| A \delta - b \right\|^2_2
\]  

(9)

where \( A_i = \Sigma_i^{-1/2} H_i, b_i = \Sigma_i^{-1/2} (z_i - h_i(X_i^0)) \).
Because the direct solution requires a large amount of computation, the QR or Cholesky decomposition method is used to accelerate the solution.

$$Q^T A = \begin{bmatrix} R \ 0 \end{bmatrix} Q^T b = \begin{bmatrix} d \ e \end{bmatrix}$$

$$\| A\delta - b \|_2^2 = \| Q^T A\delta - Q^T b \|_2^2 = \| R\delta - d \|_2^2 + \| e \|_2^2$$

(10)

However, in practical applications, with the increase in system running time, the number of factor nodes in the algorithm gradually increases. When the optimal value of the state variable is solved each time, the Jacobian matrix $J$ needs to be recalculated, and the QR decomposition is performed to obtain $R$, so as to perform the least squares optimization calculation. This batch solution method undoubtedly increases the calculation amount of the system and has a great impact on the real-time requirements of navigation and positioning.

Through comparison, the literature [35–37] found that the newly added factor only affects the adjacent node variables, so the Givens matrix is introduced, as shown in Figure 1. The sparseness of $R$ is optimized by the incremental update form and by considering different elimination orders, which reduces the calculation amount for the least squares solution and increases the real-time performance of the system.

![Figure 1. Givens matrix incremental update. (Using the Givens rotation matrix, a matrix is transformed into upper triangular form. Elements marked with X are eliminated, and elements marked with red are changed).](image)

### 3. Factor Graph Multi-Source Fusion Model Framework

The navigation system is actually a multi-variable control system, which is usually expressed in the form of state space, so that the multi-source fusion navigation system can also be modeled in the form of a factor graph model. When using the factor graph modeling method to solve navigation and positioning results such as position and attitude, the state equation and observation equation of the multi-source fusion navigation system can be expressed in the form of a factor graph.

In this paper, the IMU factor graph model is used as the main body. When sensor measurement information such as GNSS and VO is valid, the corresponding factor nodes are connected into the IMU factor graph model.

As shown in Figure 2, a factor graph architecture based on IMU/GNSS/VO is constructed, where $x$ and $a$ represent the state variable node, and $f$ represents factors, including prior factors and measurement factors. $f_{i,k}^{prior}$ and $f_{a,k}^{prior}$ represent the prior factor of the state variable, $f_{k-1,k}^{imu}$ represents the IMU pre-integration factor, $f_{k-1,k}^{bias}$ represents the bias factor, and $f_{k}^{gnss}$ and $f_{k}^{vo}$ represent the GNSS and VO measurement factors, respectively. $x = [p, v, \theta]$ represents the position, velocity, and attitude angle, respectively, and $a = [\omega_a, a_a]$ represents the gyroscope bias and accelerometer bias, respectively.
and
−pv
State node
ka
represent the GNSS and VO measurement
+−VO Factor
Bias Factor
where
f
kx
prior
IMU Factor 
Bias Factor 
X Prior 
a Prior 
GNSS Factor 
VO Factor
including prior factors and measurement factors. Prior

\( f_{01}^{\text{IMU}} \)
\( f_{12}^{\text{IMU}} \)
\( f_{23}^{\text{IMU}} \)
\( f_{k-1,k}^{\text{IMU}} \)
\( f_{x}^{\text{Prior}} \)
\( f_{a}^{\text{Prior}} \)
\( x_{0} \)
\( x_{1} \)
\( x_{2} \)
\( x_{3} \)
\( \ldots \)
\( x_{k} \)
\( a_{0} \)
\( a_{1} \)
\( a_{2} \)
\( a_{3} \)
\( \ldots \)
\( a_{k} \)

Figure 2. A multi-source fusion framework for the factor graph.

3.1. IMU Pre-Integration Factor

The IMU measurement data in one update cycle \( \Delta t = t_{k+1} - t_{k} \) are pre-integrated at
the carrier coordinate system at time \( t_{k} \), and the attitude, position, and velocity increments
are obtained as (11).

\[
\begin{aligned}
\theta_{k+1} &= \theta_{k} + H(\theta_{k})^{-1} \omega_{k}^{b} \Delta t \\
p_{k+1} &= p_{k} + v_{k} \Delta t + R_{k} a_{k}^{b} \frac{\Delta t^2}{2} \\
v_{k+1} &= v_{k} + R_{k} a_{k}^{b} \Delta t \\
H(\theta) &= \sum_{k=0}^{\infty} (-1)^{k} \theta^{k} \\
\end{aligned}
\]

where \( R_{k} = \exp( [\theta_{k}]_{\times} ) \) represents the rotation vector. \( \Delta t \) represents the pre-integration
time interval.

The factor node is represented as the error function that needs to be minimized, and
the IMU pre-integration factor node expression is as follows (12).

\[
f_{\text{IMU}}^{k,k+1}(x_{k+1}, x_{k}, a_{k}) = \left\| e_{\text{IMU}}^{k,k+1} \right\|^{2}_{\Sigma_{\text{IMU}}^{k,k+1}} = \left\| x_{k+1} - h(x_{k+1}, x_{k}, a_{k}) \right\|^{2}_{\Sigma_{\text{IMU}}^{k,k+1}}
\]

where \( h(x_{k+1}, x_{k}, a_{k}) \) represents the measurement function and \( \Sigma_{\text{IMU}}^{k,k+1} \)
represents the IMU pre-integration measurement noise covariance.

3.2. IMU Bias Factor

The IMU bias factor node expression is as follows (13).

\[
f_{\text{Bias}}^{k,k+1}(a_{k+1}, a_{k}) = \left\| e_{\text{Bias}}^{k,k+1} \right\|^{2}_{\Sigma_{\text{Bias}}^{k,k+1}} = \left\| a_{k+1} - h(a_{k}) \right\|^{2}_{\Sigma_{\text{Bias}}^{k,k+1}}
\]

where \( h(a_{k}) \) represents the measurement function and \( \Sigma_{\text{Bias}}^{k,k+1} \) represents the IMU bias
measurement noise covariance.
3.3. GNSS Factor

GNSS can provide the position and velocity information, and the measurements are expressed as (14).

\[ z_k^{\text{GNSS}} \triangleq \begin{bmatrix} p_k^{\text{GNSS}} \\ v_k^{\text{GNSS}} \end{bmatrix} = \begin{bmatrix} p_k + n_p^{\text{GNSS}} \\ v_k + n_v^{\text{GNSS}} \end{bmatrix} \]  

where \( p_k^{\text{GNSS}} \) and \( v_k^{\text{GNSS}} \) represent the GNSS position and velocity measurements, respectively. \( n_p^{\text{GNSS}} \) and \( n_v^{\text{GNSS}} \) represent the position and velocity measurement noise, respectively.

The updated GNSS factor nodes at time \( t_k \) are as follows (15).

\[ f_k^{\text{GNSS}}(x_k) = \left\| e_k^{\text{GNSS}} \right\|^2_{\Sigma_k^{\text{GNSS}}} = \left\| z_k^{\text{GNSS}} - h(x_k) \right\|^2_{\Sigma_k^{\text{GNSS}}} \]  

where \( h(x_k) \) represents the measurement function and \( \Sigma_k^{\text{GNSS}} \) represents the GNSS measurement noise covariance.

3.4. VO Factor

VO can provide the position and attitude information, and the measurements are expressed as follows (16).

\[ z_k^{\text{VO}} = \begin{bmatrix} p_k^{\text{VO}} \\ \theta_k^{\text{VO}} \end{bmatrix} = \begin{bmatrix} p_k + n_p^{\text{VO}} \\ \theta_k + n_\theta^{\text{VO}} \end{bmatrix} \]  

where \( p_k^{\text{VO}} \) and \( \theta_k^{\text{VO}} \) represent the position and attitude measurements of VO, respectively. \( n_p^{\text{VO}} \) and \( n_\theta^{\text{VO}} \) represent the position and attitude measurement noise, respectively.

The updated VO factor node at time \( t \) is as follows (17).

\[ f_k^{\text{VO}}(x_k) = \left\| e_k^{\text{VO}} \right\|^2_{\Sigma_k^{\text{VO}}} = \left\| z_k^{\text{VO}} - h(x_k) \right\|^2_{\Sigma_k^{\text{VO}}} \]  

where \( h(x_k) \) represents the measurement function and \( \Sigma_k^{\text{VO}} \) represents the VO measurement noise covariance.

4. Federated Kalman Filter and Adaptive Federated Kalman Filter (AFKF)

In order to verify the applicability and robustness of the factor graph optimization multi-source fusion algorithm proposed in this paper, the traditional Kalman filter algorithm and the adaptive federated Kalman filter algorithm (cited from Ref. [38]) used for comparison are listed below, as shown in Figures 3 and 4.

![Diagram](image-url)  

Figure 3. Federated Kalman filter.
4. Federated Kalman Filter and Adaptive Federated Kalman Filter (AFKF)

In order to verify the applicability and robustness of the factor graph optimization multi-source fusion algorithm proposed in this paper, the traditional Kalman filter algorithm and the adaptive federated Kalman filter algorithm (cited from Ref. [38]) used for comparison are listed below, as shown in Figures 3 and 4.

**Figure 3. Federated Kalman filter.**

**Figure 4. Adaptive federated Kalman filter.**

In Figure 4, the accuracy of the sub-filter $\lambda_i(k)$ is calculated by the state covariance $P_i$, as shown in (18).

$$\lambda_i(k) = \sqrt{tr(P_i(k) * P_i(k)^T)}$$  \hspace{1cm} (18)

The adaptive information sharing coefficient $\beta_i(k)$ and sub-filter precision $\lambda_i(k)$ are expressed as follows:

$$\beta_i(k) = \frac{1/\lambda_i(k)}{1/\lambda_1(k) + 1/\lambda_2(k) + \cdots + 1/\lambda_N(k)}, i = 1, 2, \ldots, N$$  \hspace{1cm} (19)

Compared with the traditional Kalman filter algorithm, the adaptive Kalman filter algorithm in Figure 4 utilizes the information-sharing coefficients for adaptive allocation, thereby improving the robustness and accuracy of the entire system.

5. Simulation Experiment Verification

We specifically show here that, as in Ref. [38], the following mathematical simulation experiments in this paper are also developed and implemented on the basis of the PSINS toolbox and completed by Professor Yan Citizen of Northwestern Polytechnical University [39].

5.1. Trajectory Simulation Settings

According to the test requirements, the trajectory of the UAV is designed, with the initial position (longitude, latitude, and altitude) being 34.812332°, 113.568645°, and 0 m, respectively, and the initial attitude (pitch, roll, and yaw) being 0°, 0°, and 0°, respectively. The initial velocity (north, east, and ground) is 0, 0, and 0 m/s, respectively. The simulated UAV motion state includes acceleration, turning, climbing, descending, decelerating, etc. The trajectory and related position, speed, and attitude state quantities are shown in Figure 5.
5. Simulation Experiment Verification

We specifically show here that, as in Ref. [38], the following mathematical simulation experiments in this paper are also developed and implemented on the basis of the PSINS toolbox and completed by Professor Yan Citizen of Northwestern Polytechnical University [39].

5.1. Trajectory Simulation Settings

According to the test requirements, the trajectory of the UAV is designed, with the initial position (longitude, latitude, and altitude) being 34.812332°, 113.568645°, and 0 m, respectively, and the initial attitude (pitch, roll, and yaw) being 0°, 0°, and 0°, respectively. The initial velocity (north, east, and ground) is 0, 0, and 0 m/s, respectively. The simulated UAV motion state includes acceleration, turning, climbing, descending, decelerating, etc. The trajectory and related position, speed, and attitude state quantities are shown in Figure 5.

Figure 5. Trajectory and parameter state simulation.

5.2. Simulation Parameter Settings

According to the trajectory requirements, the parameters of the sensors configured by the UAV are set, as shown in Table 1.
Table 1. Sensor parameter settings.

| Sensor Type | Parameter          | Value         |
|-------------|--------------------|---------------|
| IMU         | Gyro error         |                |
|             | (x-, y-, z-)       | bias          | 0.2°/h       |
|             |                    | random walk   | 0.08°/√h     |
| Accelerometer error | bias           | 100 µg       |
|             | (x-, y-, z-)       | random walk   | 20 µg/√h     |
| Frequency   |                    | 100 Hz        |
| GNSS        | Location (longitude, latitude, altitude) | [1 m; 1 m; 2 m] |
|             | Speed (north, east, down) | [0.1 m/s; 0.1 m/s; 0.1 m/s] |
|             | Frequency          | 1 Hz          |
| VO          | Location (x-, y-, z-) | [0.5 m; 0.5 m; 0.5 m] |
|             | Attitude (pitch-, yaw-, roll-) | [0.5°; 0.5°; 0.5°] |
|             | Frequency          | 1 Hz          |

5.3. Simulation Scene Settings

In this paper, the simulation scene is set in combination with the challenging environment faced by the UAV in actual flight, and the different state periods in which the sensor is prone to measurement errors are designed. The measurement errors and corresponding time periods produced by different sensors are shown in Table 2.

Table 2. Measurement errors of different sensors at different time periods (40~170 s, 20 times the $\Sigma_{k}^{\text{GNSS}}$ gross error is added to the GNSS positioning measurement. In the range of 410~500 s, 10 times the $\Sigma_{k}^{\text{VO}}$ gross error is added to the VO positioning measurement).

| Sensor Type | 40~170 s | 410~500 s |
|-------------|----------|-----------|
| VO          |          | 20 $\times$ $\Sigma_{k}^{\text{GNSS}}$ |
| GNSS        |          | 10 $\times$ $\Sigma_{k}^{\text{VO}}$ |

5.4. Experimental Results and Discussion

To verify the adaptability and robustness of the factor graph optimization algorithm, on the basis of a comparative analysis of the accuracy and time efficiency of different sliding windows, the results of the factor graph optimization algorithm, the traditional federated Kalman filtering algorithm, and the adaptive federated Kalman filtering algorithm were obtained for comparison and analysis.

5.4.1. Comparative Analysis of FGO Sliding Window Size

The calculation time and accuracy of the FGO algorithms in this paper are tested. The test environment is Ubuntu 18.04, MATLAB 2018a; the test platform is configured using a 1.99 GHz, Intel(R) Core (TM) i7-8550U CPU. The test software is MATLAB 2018a, and the gtsam [40,41] library is used. By exploiting Cholesky decomposition and by constructing NonlinearFactorGraph using the Gaussian mixture model, the factor graph optimization model is built.

In order to compare and to analyze the relationship between real-time performance and accuracy, Figure 6 shows the change in position accuracy of different sliding window sizes, and Table 3 shows the comparison results of the position errors of different window size factor maps and the time used for the single-step execution.
In order to compare and to analyze the relationship between real-time performance and the time used for the single-step execution, the accuracy of position error and the time used for the single-step execution for different window sizes, including 1, 10, 30, 100, and 391 s, are shown in Table 3.

Table 3. Comparison results of position accuracy and the time used for the single-step execution for different window sizes.

| Windows Size (s) | 1      | 10     | 30     | 100    | 391    |
|------------------|--------|--------|--------|--------|--------|
| Position Error (m) | 0.24   | 0.17   | 0.12   | 0.10   | 0.09   |
| The Used Time (s)  | 5.6971 × 10⁻² | 6.9302 × 10⁻² | 7.4211 × 10⁻² | 9.8083 × 10⁻² | 1.23402 × 10⁻¹ |

As shown in Table 3, the accuracy continues to improve as the sliding window size increases. This is because the historical information is increased after the window is increased, especially for the window of 391, which is equivalent to batch optimization. Batch optimization considers global information, so that the accuracy is the highest. However, as the window increases, the usage time also increases, which is a major challenge for real-time performance. For a window larger than 30 s (100 s, 391 s), the positioning accuracy is not much improved, but the time consumption is significantly increased. For example, a single-step operation with a window of 391 takes twice as long as a window of 30 s, which exceeds the real-time requirement. Finally, considering the balance of time and accuracy, 30 s is chosen as the FGO sliding window size in the following.

5.4.2. Track Comparison Results

In this study, three algorithms (FKF, AFKF, and FGO) were used for simulation experiments; the simulation trajectories, attitude, speed, and position errors were compared and analyzed; and the applicability of the FGO algorithm was analyzed.

First, the simulation comparison of the trajectory is shown in Figure 7, below. Black lines represent true trajectories, red lines represent FKF, green lines represent AFKF, and blue lines represent FGO. It can be seen from Figure 7 that all three simulation algorithms can track the real trajectory, even in the two periods of sensor failure set in this paper. However, in general, the factor graph algorithm is closer to the real trajectory, followed by AFKF, and finally, FKF. This is because the factor graph method can realize the plug-and-play function simply by adding or subtracting the factor nodes corresponding to the navigation sensor. Compared with the adaptive federated Kalman filtering algorithm, the factor graph algorithm has better flexibility and scalability.
5.4.3. Error Analysis and Precision Statistics

In order to further compare and analyze the effects of the three algorithms on the state quantity in detail, the following attitude errors (pitch angle, roll angle, and yaw angle), speed errors (north speed, east speed, and down speed), and position errors (latitude, longitude, and height) are compared, as shown in Figures 8–10. The red line represents the FKF error result, the green line represents the AFKF error result, and the blue line represents the FGO error result.

Figure 7. Comparison of different algorithm trajectories.

Figure 8. Comparison of attitude errors of different algorithms.
parameter of FGO is also small. Therefore, the FGO algorithm has the highest positioning accuracy, followed by AFKF, and finally FKF. Overall, the attitude of FGO is stronger stability and smaller errors in position, velocity, and attitude.

In order to quantitatively analyze the errors of each navigation parameter, the absolute mean error (AME), root mean square error (RMSE), and standard deviation (STD) of the three algorithms (FKF, AFKF, and FGO) are compared, as shown in Table 4. From the RMSE and AME data in Table 4, it can be seen that the state estimation result of FGO is the closest to the true value, and it can be seen from the STD data that the discrete degree of each state parameter of FGO is also small. Therefore, the FGO algorithm has the highest positioning accuracy, followed by AFKF, and finally FKF. Overall, the attitude of FGO is 8 arc minutes higher than that of FKF, and the speed is not much different, which is increased by 0.01 m/s, and the position is increased by 0.18 m.
Table 4. Error statistics of different algorithms.

| Error | Dlat (m) | Dlon (m) | dH (m) | dVN (m/s) | dVE (m/s) | dVD (m/s) | Dpith (°) | Dyaw (°) | Droll (°) |
|-------|----------|----------|--------|-----------|-----------|-----------|-----------|----------|-----------|
| FKF   | AME      | 0.25     | 0.37   | 0.27      | 0.02      | 0.02      | 0.01      | 14.55    | 17.66     | 20.33     |
|       | RMSE     | 0.31     | 0.44   | 0.36      | 0.02      | 0.02      | 0.02      | 20.91    | 20.70     | 21.07     |
|       | STD      | 0.18     | 0.22   | 0.23      | 0.06      | 0.06      | 0.02      | 19.47    | 19.18     | 15.53     |
| AFKF  | AME      | 0.18     | 0.19   | 0.09      | 0.01      | 0.01      | 0.00      | 11.06    | 8.98      | 10.43     |
|       | RMSE     | 0.25     | 0.25   | 0.12      | 0.02      | 0.02      | 0.01      | 13.86    | 10.93     | 11.67     |
|       | STD      | 0.17     | 0.15   | 0.08      | 0.03      | 0.05      | 0.02      | 13.90    | 10.98     | 6.68      |
| FGO   | AME      | 0.08     | 0.11   | 0.09      | 0.01      | 0.01      | 0.01      | 9.33     | 11.59     | 6.81      |
|       | RMSE     | 0.10     | 0.14   | 0.11      | 0.01      | 0.01      | 0.01      | 12.31    | 13.52     | 8.31      |
|       | STD      | 0.07     | 0.08   | 0.06      | 0.01      | 0.01      | 0.01      | 12.34    | 11.60     | 3.99      |

5.4.4. Statistical Analysis of Multiple Experiments

To further compare the accuracies of the three algorithms, this study performed 20 groups of Monte Carlo simulations to simulate the real environment. The noise, trajectory, and speed of each setting are different. The mean absolute errors (MAEs) of the position errors for 20 groups of experiments are listed in Table 5, and the MAEs are shown in Figure 11.

Table 5. The MAE values of position errors (m) in 20 groups of Monte Carlo simulation experiments for the three algorithms (FKF, AFKF, and FGO).

| Number | FKF    | AFKF   | FGO    |
|--------|--------|--------|--------|
| 1      | 0.3144 | 0.2138 | 0.1444 |
| 2      | 0.2862 | 0.1563 | 0.0979 |
| 3      | 0.3044 | 0.1502 | 0.1101 |
| 4      | 0.3372 | 0.1843 | 0.0656 |
| 5      | 0.3049 | 0.1745 | 0.1198 |
| 6      | 0.2702 | 0.1828 | 0.1059 |
| 7      | 0.2756 | 0.1831 | 0.0595 |
| 8      | 0.2962 | 0.1154 | 0.0954 |
| 9      | 0.2955 | 0.1760 | 0.0807 |
| 10     | 0.2753 | 0.1571 | 0.0975 |
| 11     | 0.3067 | 0.1370 | 0.0867 |
| 12     | 0.3393 | 0.1390 | 0.0885 |
| 13     | 0.2938 | 0.1982 | 0.0632 |
| 14     | 0.2952 | 0.1355 | 0.0845 |
| 15     | 0.2353 | 0.1227 | 0.1172 |
| 16     | 0.2781 | 0.1662 | 0.1465 |
| 17     | 0.2595 | 0.2201 | 0.0594 |
| 18     | 0.2897 | 0.1606 | 0.1655 |
| 19     | 0.3171 | 0.1690 | 0.1400 |
| 20     | 0.3470 | 0.1317 | 0.0993 |
The specific structure and sensor configuration are shown in Figure 12, and the specific parameters and calibration data of the sensor are shown in Table 6.

As shown in Figure 11 and Table 2, the accuracy of FGO is significantly better than that of the other two algorithms. The average position error accuracies of the 20 groups of experiments were calculated separately, and the errors of the three algorithms are obtained as: 0.3013, 0.1598, and 0.0942 m. Compared with AFKF, the accuracy of FGO is increased by two-fold, and compared with FKF, the accuracy is increased by three-fold. The discussion and analysis of the above results further prove that the FGO algorithm proposed in this paper has a high accuracy and a good robustness, and the algorithm can be applied to complex environments.

6. Dataset Validation

In order to further verify the applicability of the FGO algorithm, the following is verified by the EuRoC dataset [42]. The EuRoC dataset is a public dataset released by the Zurich University of Technology in 2016. The data collection platform is the Asctec Firefly, a rotary-wing drone, equipped with cameras, IMU, and visual motion capture systems. The specific structure and sensor configuration are shown in Figure 12, and the specific parameters and calibration data of the sensor are shown in Table 6.

Figure 11. The MAEs of position errors (m) in 20 groups of Monte Carlo simulation experiments for the three algorithms (FKF, AFKF, and FGO).

Figure 12. Asctec Firefly drone and sensor configuration parameters.
Table 6. Calibration values of internal and external parameters of the sensor (reprinted/adapted with permission from Ref. [42]).

| Parameter                     | Value                                                        |
|-------------------------------|--------------------------------------------------------------|
| Resolution                   | [752, 480] pix                                              |
| intrinsics                    | 458.654 0 367.215 0 457.296 248.375                         |
| distortion_coefficients       | [−0.28340811 0.07395907 0.00019359 1.76187114 × 10⁻⁵]         |
| gyroscope_noise_density       | 1.6968 × 10⁻⁴ rad/s/√Hz                                      |
| gyroscope_random_walk         | 1.9393 × 10⁻⁵ rad/s²/√Hz                                     |
| accelerometer_noise_density  | 2.0000 × 10⁻³ m/s²/√Hz                                      |
| accelerometer_random_walk     | 3.0000 × 10⁻³ m/s³/√Hz                                      |
| Camera-IMU extrinsics         | 0.0 0.0 0.0 1.0                                              |

The EuRoC data include both types of data: Industrial Machine Hall and Vicon Room. In order to fully verify the applicability of the algorithm in different scenarios and conditions, we chose three types of data: simple (MH_01_easy), medium (MH_03_medium), and more difficult (V1_03_difficult), for analysis and verification.

In order to verify the multi-source fusion effect of three sensors, IMU, VO, and GNSS, the GNSS data are simulated and generated according to the existing data (the location data of state_groundtruth_estimate0 is added with 0.5 m error, speed is added with 0.1 m/s error, and the frequency is 10 Hz). The VO data are calculated using ORB-SLAM2, and the GNSS and VO data are aligned for the following use. In order to verify the FGO algorithm in complex scenarios, errors are added to different sensors at different time periods, as shown in Table 7.

Table 7. Different sensor simulation error parameter values.

| Scenes                  | Sensor Type | Value                  | Period   |
|-------------------------|-------------|------------------------|----------|
| MH_01_easy              | GNSS        | 20 × Σ³GNSS            | 40–100 s |
|                         | VO          | 10 × Σ³VO              | 140–180 s|
| MH_03_medium            | GNSS        | 20 × Σ³GNSS            | 20–40 s  |
|                         | VO          | 10 × Σ³VO              | 60–90 s  |
| V1_03_difficult         | GNSS        | 20 × Σ³GNSS            | 20–40 s  |
|                         | VO          | 10 × Σ³VO              | 70–90 s  |

6.1. MH_01_easy Scene

The MH_01_easy scene has a running length of 80.6 m and a duration of 182 s. It is a good texture and a bright scene. Figure 13 shows the 3D position graph and the 2D position graph of different algorithms for the MH_01_easy scene. The black line represents ground truth, the red line represents the state estimation result calculated by the FKF algorithm, the green line represents the state estimation result computed by the AFKF algorithm, and the blue line represents the state estimation result calculated by the FGO algorithm.
As can be seen from Figure 15, the trajectories obtained by the three algorithms can all track the ground truth. The black, red, green, and blue lines, respectively, represent ground truth, FKF state estimation trajectory, AFKF state estimation trajectory, and FGO state estimation trajectory. Figure 15 shows a comparison diagram of the 3D position and 2D position of different algorithms for the MH_03_medium scene. The black, red, green, and blue lines, respectively, represent ground truth, FKF state estimation error, AFKF state estimation error, and FGO state estimation error. FKF shows different changes according to the changes in different sensor errors in different time periods. The position error of FGO does not fluctuate, and the error value is small.

Figure 13. Comparison of 3D position and 2D position of different algorithms in MH_01_easy scene.

The 3D position and 2D position trajectories of the MH_01_easy scene are shown in Figure 13. Compared with AFKF and FKF, FGO is closer to the real trajectory, and has a smaller position error. However, the FGO trajectory appears to be locally unsmoothed, because the factor is automatically added according to different environments (sensor gross errors).

Figure 14 is the position error comparison diagram of the MH_01_easy scene. Red represents the FKF state estimation error, green represents the AFKF state estimation error, and blue represents the FGO state estimation error. FKF shows different changes according to the changes in different sensor errors in different time periods. The position error of FGO does not fluctuate, and the error value is small.

Figure 14. Comparison of position errors of different algorithms in MH_01_easy scene.

6.2. MH_03_MEDIUM SCENE

This scene has a running length of 130.9 m and a time of 132 s, and it is a fast-motion and bright scene. Similar to the MH_01_easy scene, Figure 15 shows a comparison diagram of the 3D position and 2D position of different algorithms for the MH_03_medium scene. The black, red, green, and blue lines, respectively, represent ground truth, FKF state estimation trajectory, AFKF state estimation trajectory, and FGO state estimation trajectory. As can be seen from Figure 15, the trajectories obtained by the three algorithms can all track the ground truth.

Figure 15. Comparison of position errors of different algorithms in MH_03_medium scene.
As shown in Figure 16, the error results for FKF, AFKF, and FGO are represented by red, green, and blue lines, respectively. Compared with the MH_01_easy scene, the position errors of the three algorithms in this scene are larger. However, compared with FKF and AFKF, FGO has a higher position accuracy.

6.3. V1_03_Difficult Scene

The scene is a fast-motion and motion-blur scene, with a total running length of 79.0 m and a running time of 105 s. Similar to the MH_01_easy scene and the MH_03_medium scene, Figure 17 shows the 3D and 2D position comparison of different algorithms in the V1_03_difficult scene. The black, red, green, and blue lines represent ground truth, FKF state estimation trajectory, AFKF state estimation trajectory, and FGO state estimation trajectory, respectively. Figure 18 shows the error results for FKF, AFKF and FGO. Compared with the first two scenarios, the trajectories obtained by the FKF, AFKF, and FGO algorithms in this scenario have a certain deviation from the ground truth, but the effect of FGO is still better than that of AFKF and FKF.
To sum up the comparison, the data of the three scenarios are compared and analyzed, and the comparison results of the position error and the attitude errors of different algorithms in different scenarios are shown in Table 8. For a simple scene (MH_01_easy scene), FGO has obvious advantages. The algorithm has higher overall accuracy, and the accuracy remains relatively stable. As the complexity of the scene increases from the MH_01_easy scene to the V1_03_difficult scene, the overall accuracies of the FKF, AFKF, and FGO algorithms decrease, but the accuracy of FGO is still higher than those of AFKF and FKF. Combining the three scenarios, the position accuracy is obviously improved. Compared with AFKF and FKF, the positioning accuracy of FGO is improved by 1.5–2-fold. This positioning accuracy result is worse than the simulation data. This is because the difficulty of the scene increases, resulting in data with stronger dynamics, fewer visual feature points, and a larger range of motion.
Table 8. Accuracy comparison results of different algorithms in different scenarios.

| Experimental Scene      | Algorithm | x   | y   | z   |
|-------------------------|-----------|-----|-----|-----|
| MH_01_easy              | FKF       | 0.07| 0.07| 0.07|
| MH_01_easy              | AFKF      | 0.06| 0.06| 0.05|
| MH_01_easy              | FGO       | 0.04| 0.04| 0.03|
| MH_03_medium            | FKF       | 0.09| 0.08| 0.07|
| MH_03_medium            | AFKF      | 0.07| 0.06| 0.05|
| MH_03_medium            | FGO       | 0.05| 0.05| 0.03|
| V1_03_difficult         | FKF       | 0.12| 0.14| 0.11|
| V1_03_difficult         | AFKF      | 0.10| 0.11| 0.08|
| V1_03_difficult         | FGO       | 0.09| 0.09| 0.05|

7. Conclusions

To improve the reliability and robustness of UAV autonomous navigation and positioning in complex scenarios, we used a novel multi-source fusion algorithm framework for autonomous navigation and localization. The main innovation is that the multi-source fusion framework established in this paper considers the IMU pre-integration factor and IMU bias factor at the same time, and iSAM is used to solve it. The requirements of positioning accuracy and real-time performance can be met at the same time. In addition, compared with previous studies, the selection of the sliding window size was first considered in this paper. Compared with the traditional Kalman filter and the adaptive Kalman filter, the results show that FGO has the highest accuracy, followed by AFKF, and the worst is FKF. Finally, three scenarios of EuRoC datasets with different difficulties were compared and analyzed, which further verifies the usability and robustness of FGO. Combined with the mathematical simulation experiments and data set verification, compared with the other two algorithms, the position accuracy of FGO is improved by 1.5–2-fold.

In summary, the FGO algorithm in this paper can significantly improve the accuracy and tolerance of the navigation system in complex environments. However, so far, most factor graph algorithms are only in the experimental simulation stage, are rarely applied to practical systems [43–45], and a lot of theoretical work and engineering practice are needed in the next step. The multi-sensor fusion algorithm based on a factor graph in this paper is based on a loosely coupled structure. In the next research, we plan to establish a tightly coupled fusion framework to further improve the navigation accuracy.

Author Contributions: J.D., S.L., X.H., Z.R. and X.Y. conceived and designed this study; J.D., S.L. and X.H. performed the experiments; J.D. wrote the paper; S.L., Z.R. and X.Y. reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the Science and Technology Department of Henan Province through the project on Research on Key Technologies for Fully Autonomous Operation of Multi-rotor Agricultural Plant Protection UAVs (No. 222102110029), the project on Research on Collaborative Assembly Technology of Aviation Complex Parts Based on Multi-Agent Reinforcement Learning (No. 222102220095), and the project on intelligent plant protection drones (No. 162102216237). In addition, the research was supported by Henan Provincial Department of Education through the project on UAV Positioning Navigation and Autonomous Control Technology with Missing Satellite Navigation Signals (No. 212102210509).

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. Sun, Y.; Song, L.; Wang, G. Overview of the development of foreign ground unmanned autonomous systems in 2019. Aerodyn. Missile J. 2020, 1, 30–34.

2. Zhang, T.; Li, Q.; Zhang, C.S.; Liang, H.W.; Li, P.; Wang, T.M.; Li, S.; Zhu, Y.L.; Wu, C. Current Trends in the Development of Intelligent Unmanned Autonomous Systems. Unmanned Syst. Technol. 2018, 68–85. [CrossRef]

3. Guo, C. Key Technical Research of Information Fusion for Multi-source Integrated Navigation System. Ph.D. Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2018.

4. Tang, L.; Tang, X.; Li, B.; Liu, X. A Survey of Fusion Algorithms for Multi-source Fusion Navigation Systems. GNSS Word China 2018, 43, 39–44.

5. Wang, Q.; Cui, X.; Li, Y.; Ye, F. Performance Enhancement of a USV INS/CNS/DVL Integration Navigation System Based on an Adaptive Information Sharing Factor Federated Filter. Sensors 2017, 17, 239. [CrossRef] [PubMed]

6. Xu, X.; Pang, F.; Ran, Y.; Bai, Y.; Zhang, L.; Tan, Z.; Wei, C.; Luo, M. An Indoor Mobile Robot Positioning Algorithm Based on Adaptive Federated Kalman Filter. IEEE Sens. J. 2021, 21, 23098–23107. [CrossRef]

7. Frank, D.; Michael, K. Factor Graphs for Robot Perception; Foundations and Trends® in Robotics Series; Now Publishers: Delft, The Netherlands, 2017; Volume 6, pp. 1–139.

8. Zhu, X.; Chen, S.; Jiang, C. Integrated navigation based on graph optimization method and its feasibility. Electron. Opt. Control 2019, 26, 66–70.

9. Wang, M.; Li, Y.; Feng, G. Key technologies of GNSS/INS/VO deep integration for UGV navigation in urban canyon. In Proceedings of the 2017 11th Asian Control Conference, Gold Coast, Australia, 7–20 December 2017; pp. 2546–2551.

10. Xu, H.; Lian, B.; Liu, S. Multi-source Combined Navigation Factor Graph Fusion Algorithm Based on Sliding Window Iterative Maximum Posterior Estimation. J. Mil. Eng. 2019, 40, 807–819.

11. Levinson, J.; Montemerlo, M.; Thrun, S. Map-Based Precision Vehicle Localization in Urban Environments. In Robotics: Science & Systems; Georgia Institute of Technology: Atlanta, GA, USA, 2007.

12. Levinson, J.; Thrun, S. Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In Proceedings of the IEEE International Conference on Robotics & Automation, Anchorage, AK, USA, 3–7 May 2010.

13. Ding, W.; Hou, S.; Gao, H. LiDAR Inertial Odometry Aided Robust LiDAR Localization System in Changing City Scenes. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020.

14. Pfeifer, T.; Protzel, P. Robust Sensor Fusion with Self-Tuning Mixture Models. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.

15. Wang, H.; Zeng, Q.; Liu, J. Research on the key technology of UAV of all source position navigation based on factor graph. Navig. Control 2017, 16, 1–5.

16. Chen, M.; Xiong, Z.; Liu, J. Distributed cooperative navigation method of UAV swarm based on factor graph. J. Chin. Inert. Technol. 2020, 28, 456–461.

17. Tang, C.; Zhang, L.; Lian, B. Cooperation factor map of co-location aided single satellite navigation algorithm. Syst. Eng. Electron. 2017, 39, 1085–1090.

18. Gao, J.; Tang, X.; Zhang, H. Vehicle INS/GNSS/OD integrated navigation algorithm based on factor graph. Syst. Eng. Electron. 2018, 40, 2547–2554.

19. Indelman, V.; Williams, S.; Kaess, M. Information Fusion in Navigation Systems via Factor Graph Based Incremental Smoothing. Robot. Auton. Syst. 2013, 61, 721–738. [CrossRef]

20. Xu, J.; Yang, G.; Sun, Y. A multi-sensor information fusion method based on factor graph for integrated navigation system. IEEE Access 2021, 9, 12044–12054. [CrossRef]

21. Wei, X.; Li, J.; Zhang, D. An improved integrated navigation method with enhanced robustness based on factor graph. Mech. Syst. Signal Process. 2021, 155, 107565. [CrossRef]

22. Yang, S.; Tan, J.; Chen, B. Robust Spike-Based Continual Meta-Learning Improved by Restricted Minimum Error Entropy Criterion. Entropy 2022, 24, 455. [CrossRef] [PubMed]

23. Yang, S.; Linares-Barranco, B.; Chen, B. Heterogeneous Ensemble-Based Spike-Driven Few-Shot Online Learning. Front. Neuro. 2022, 16, 850932. [CrossRef] [PubMed]

24. Liu, J.; Wei, Z.; Li, Z. SAM: A Self-adaptive Attention Module for Context-Aware Recommendation System. arXiv 2021, arXiv:2110.00452. [CrossRef] [PubMed]

25. Yang, S.; Gao, T.; Wang, J.; Deng, B.; Landsell, B.; Linares-Barranco, B. Efficient Spike-Driven Learning with Dendritic Event-Based Processing. Front. Neurosci. 2021, 15, 601109. [CrossRef] [PubMed]

26. Zeng, Q.; Chen, W.; Liu, J. An improved multi-sensor fusion navigation algorithm based on the factor graph. Sensors 2017, 17, 641. [CrossRef] [PubMed]

27. Yao, Z.; Liu, Y.; Guo, J. Multi-source heterogeneous information fusion algorithm for autonomous navigation based on factor graph. Electron. Meas. Technol. 2021, 3, 130–134.

28. Luo, Z.; Chen, S.; Wang, G. A review of factor graph algorithms for multi-source fusion navigation systems. Navig. Control 2021, 20, 9–16.

29. Zhang, J.; Wang, X.; Deng, Z. An asynchronous information fusion positioning algorithm based on factor graph. Missiles Space Veh. 2019, 3, 89–95.
30. Zhao, W.; Meng, W.; Chi, Y.; Han, S. Factor Graph based Multi-source Data Fusion for Wireless Localization. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Doha, Qatar, 3–6 April 2016; pp. 592–597.

31. Wu, X.; Xiao, B.; Wu, C. Factor graph based navigation and positioning for control system design: A review-ScienceDirect. Chin. J. Aeronaut. 2021, 35, 25–39. [CrossRef]

32. Frey, B.J.; Kschischang, F.; Loeliger, H. Factor graphs and algorithms. In Proceedings of the 35th Allerton Conference on Communications, Control, and Computing, Monticello, IL, USA, 22–24 September 1999; pp. 666–680.

33. Koetter, R. Factor graphs and iterative algorithms. In Proceedings of the 1999 Information Theory and Networking Workshop, Metsovo, Greece, 27 June–1 July 1999.

34. Christian, S.; Lance, P. Factor Graphs. In Trellis and Turbo Coding; IEEE: Piscataway, NJ, USA, 2004; pp. 227–249. [CrossRef]

35. Kaess, M.; Ranganathan, A.; Dellaert, F. Incremental Smoothing and Mapping. IEEE Trans. Robot. 2008, 24, 1365–1378. [CrossRef]

36. Kaess, M.; Johannsson, H.; Roberts, R. iSAM2: Incremental Smoothing and Mapping with Fluid Relinearization and Incremental Variable Reordering. In Proceedings of the IEEE International Conference on Robotics & Automation, Shanghai, China, 9–13 May 2011.

37. Kaess, M.; Ila, V.; Roberts, R. The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping. In Algorithmic Foundations of Robotics IX-Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics; WAFR: Singapore, 2010.

38. Dai, J.; Hao, X.; Liu, S.; Ren, Z. Research on UAV Robust Adaptive Positioning Algorithm Based on IMU/GNSS/VO in Complex Scenes. Sensors 2022, 22, 2832. [CrossRef]

39. Yan, G.; Deng, Y. Review on practical Kalman filtering techniques in traditional integrated navigation system. Navig. Position Timing 2020, 7, 50–64.

40. Dellaert, F. Factor Graphs and GTSAM: A Hands-on Introduction; Georgia Institute of Technology: Atlanta, GA, USA, 2012.

41. Lange, S.; Nderhauf, S.U.; Protzel, P. Incremental smoothing vs. filtering for sensor fusion on an indoor UAV. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 1773–1778.

42. Burri, J.; Nikolic, P.; Gohl, T.; Schneider, J.; Rehder, S.; Omari, M.; Achtelik, M.W.; Siegwart, R. The EuRoC micro aerial vehicle datasets. Int. J. Robot. Res. 2016, 35, 1157–1163. [CrossRef]

43. Wen, W.; Pfeifer, T.; Bai, X. Factor graph optimization for GNSS/INS integration: A comparison with the extended Kalman filter. Navigation 2021, 68, 315–331. [CrossRef]

44. Wen, W.; Kan, Y.; Hsu, L. Performance Comparison of GNSS/INS Integration Based on EKF and Factor Graph Optimization. In Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), Miami, FL, USA, 16–20 September 2019.

45. Shan, G.; Park, B.H.; Nam, S.H. A 3-dimensional triangulation scheme to improve the accuracy of indoor localization for IoT services. In Proceedings of the 2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Canada, 24–26 August 2015.