Omni-swarm: An Aerial Swarm System with Decentralized Omni-directional Visual-Inertial-UWB State Estimation
Hao Xu, Yichen Zhang, Boyu Zhou, Luqi Wang, Shaojie Shen

Abstract—The collaboration of unmanned aerial vehicles (UAVs), also known as aerial swarm, has become a popular research topic for its practicality and flexibility in plenty of scenarios. However, one of the most fundamental components for autonomous aerial swarm systems in GPS-denied areas, the robust decentralized relative state estimation, remains to be an extremely challenging research topic. In order to address this research niche, the Omni-swarm, an aerial swarm system with decentralized Omni-directional visual-inertial-UWB state estimation, which features robustness, accuracy, and global consistency, is proposed in this paper. We introduce a map-based localization method using deep learning tools to perform relative localization and re-localization within the aerial swarm while achieving the fast initialization and maintaining the global consistency of state estimation. Furthermore, to overcome the sensors’ visibility issues with the limited field of view (FoV), which severely affect the performance of the state estimation, Omni-directional sensors, including fisheye cameras and ultra-wideband (UWB) sensors, are adopted. The state estimation module, together with the planning and the control modules, is integrated on the aerial system with Omni-directional sensors to attain the Omni-swarm, and extensive experiments are performed to verify the validity and examine the performance of the proposed framework. According to the experiment result, the proposed framework can achieve centimeter-level relative state estimation accuracy while ensuring global consistency.

I. INTRODUCTION

For any aerial robotic system, the state estimation for positions and attitudes according to the perception data is crucial for the control of the system, meanwhile provide a solid foundation for conducting further higher-level work, such as path planning [1], mapping [2], and can be adopted in numerous amount of applications, such as cinematography [3], search & rescue [4], and drone racing [5]. The state estimation techniques for single drone systems in GPS-denied areas are generally well-developed using visual-inertial odometry(VIO) [6]–[8]. However, the state estimation problem becomes much more complicated in a multiple-drone system, also known as an aerial swarm.

In an aerial swarm, the state estimation is extremely critical to ensure flight safety in complex missions with multiple cooperative drones. Each drone needs to not only estimate its ego state but also obtain the relative poses with other drones. To date, the vast majority of aerial swarm research has been based on motion capture systems [9], GPS [10] and RTK-GPS [11] systems, significantly limiting the application of drone swarms in the real world. As stated before, although the ego state estimation techniques are comparatively mature, the relative state estimation for aerial swarms in GPS-Denied areas remains a research niche. Besides, to achieve flexibility and robustness, getting rid of the cumbersome external localization devices such as UWB anchor or motion capture system while remaining decentralized is essential. In addition, to deploy the relative state estimation module on the drones, the restricted sensing and computing capabilities due to the size, weight, and power (SwAp) limitations need to be taken into account.

Despite that the state-of-the-art relative state estimation methods for aerial swarms [12]–[16] have been tackling these problems, they still suffer from the following issues:

1) Complicated initialization. The previous methods require large motions and sufficient data associations of the detection results to initialize the system. The complex initialization procedure may engender large errors and even failures in initialization, causing severe safety issues and even crashes.

2) Restricted field of view (FoV). Due to the FoV limitation, when the a drone flies out of the FoV of the others, it is hard to estimate its relative pose. In extreme situations, when all the drones of the aerial swarm are out of the FoV of each other, all the relative states becomes unobservable and impossible to estimate.

3) Lack of global consistency. The estimated poses drift along with the VIO, causing consistency issues.

4) Insufficient accuracy. The estimated position errors of the previous works are generally around the half-meter
level, meaning that the swarm systems can be scarcely adopted in confined indoor space or close formation scenarios.

In order to address these challenges, in this paper, we propose the Omni-swarm, an aerial swarm system with decentralized omni-directional visual-inertial-UWB state estimation. Built-up from the previous work [12], the proposed Omni-swarm extends the relative state estimation with a multi-drone map-based localization and re-localization method inspired by the loop closure [17], [18]. With the maps in the visual databases containing the landmarks and features generated by deep-learning, the relative poses between different drones and different timestamps can be estimated. With the extracted relative poses, the state estimation is able to ensure global consistency and achieve fast initialization. Moreover, the framework expands the FoV by adopting stereo fisheye cameras and ultra-wideband (UWB) sensors to become truly Omni-directional. The state estimation method is functional in known and unknown, narrow indoor and open outdoor environments with accuracy enhancements compared with our previous work.

To be detailed, as shown in Fig. 3c, with the help of fisheye cameras, the available horizontal field of view for the visual detector extends from 90 degrees to 360 degrees, and the vertical field of view extends from 60 degrees to 360 degrees, which can be regarded as omni-directional.

Inspired by the loop closure method for single drone [17], [18], we introduce a multi-drone map-based localization method in this paper. The multi-drone map-based localization method, which includes relative localization and re-localization, is utilized to ensure the estimator’s global consistency and consistency across the swarm. The maps, which contain the landmarks and with deep-learning generated features, are stored in visual databases. We can extract the relative pose between different drones or the same drone at the different timestamps by utilizing the map. The proposed multi-drone map-based localization method brings several new features:

1) Re-localization. All state estimations in the previous work were performed under the local frame of the VIO, and drift is inevitable due to the nature of the VIO itself. With the help of multi-drone map-based localization, we are able to eliminate these drifts and thus guarantee the global consistency of the estimation results and the consistency across the swarm.

2) Fast initialization. In contrast to previous methods that require motion to initialize the state estimate, we have the capability to directly extract the relative poses among the drones with the multi-drone map-based localization, thus quickly initializing the relative state estimate of the swarm even if the aerial swarm is static on the ground.

3) Accuracy enhancements. The multi-drone map-based localization implies more data input, which can improve the accuracy of state estimation.

We proposed a visual-inertial-UWB fusion method for relative state estimation in our previous work [12]. This method has the advantages of both vision-based [13], [14] and UWB-based methods [15], [16] for relative state estimation. However, the method still suffers from the initialization issue, restricted FoV issue, and global consistency issue. Although the previous method works in the non-line-of-sight case, the relative estimation accuracy will deteriorate. In this paper, the initialization and global consistency issues are addressed by introducing the multi-drone map-based localization. The non-line-of-sight issue is fixed by using omnidirectional cameras instead of pinhole cameras.

Based on the integrated complete aerial swarm system containing the drone, communication network, software management, and ground user interface, extensive experiments are conducted to validate the Omni-swarm system and the analysis of the proposed relative state estimation method is performed to examine the accuracy.

We further designed a distributed multi-agent planning protocol and conducted experiments on inter-drone obstacle avoidance between indoor and outdoor drones to demonstrate the real-time performance and practical application capabilities of our state estimation method and aerial swarm system.

To briefly summarize, the proposed Omni-swarm has the following feature:

1) Decentralized. Our algorithms runs on each of the drones individually in stead of using a central server.

2) Omni-directional. The omni-directional sensors without the FoV limitation are adopted.

3) High-accuracy. The relative state estimations are able to reach centimeter-level accuracy in real-world experiments.

4) Global consistency. The global consistency of the estimated state of the aerial swarm is guaranteed by the map based relative localization and re-localization.

5) Robustness. Our approach has been developed to compensate possible temporary signal loss and partial sensor failures. In addition, By incorporating outlier rejection, robustness can be ensured.

6) Plug and play. The system allows the temporary join or exit of the swarm agents.

II. RELATED WORKS

In previous years, researchers on aerial swarms are more likely to focus on planning and control, and the relative state estimation is likely leaped by adopting external positioning systems, including GPS [10], motion capture system [9] or UWB modules with anchers [19], which are all centralized system, requiring bulky external devices. Obviously, GPS can only work in open outdoor environments. However, real-world applications require the aerial swarm to work in different environments, including urban areas and indoor environments.

Recently, we have seen various relative state estimation methods for aerial swarms that make it possible to deploy autonomous aerial swarms in GPS-denied aerial swarms [12], [13], [15]. Existing methods for relative state estimation can be divided into three categories:

---

1In practical, we may disable some of useless FoV to save the computation resources depends on the application.
1) UWB-Odometry fusion methods [15], [16].
2) Visual object detection relative localization methods [13], [14].
3) Visual feature based relative localization methods [20], [21].

However, there are still some problems affecting the deployment of relative state estimation in a real aerial swarm. UWB-based methods [15] fuse visual odometry and UWB distance measurements to achieve relative state estimation. These methods have the ability to estimate the relative state of an aerial swarm in a non-line-of-sight situation but provide accuracy only at the meter to decimeter level, which greatly limits the practical application of the method in a narrow environment. An additional drawback of UWB-based methods is that their methods require a certain amount of motion for initialization, which may cause insecurity in a narrow environment. Meanwhile, relative localization may be unobservable in some cases, e.g., parallel flights of aerial swarms.

Visual detection methods [13], [14] deliver centimeter-level accuracy. However, this method requires that other drones remain in line-of-sight at all times. On the other hand, the association of the data from the detection results is also an issue. Paper [14] proposed a Coupled Probabilistic Data Association Filter (CPDAF) method to associate the detection result and estimate the relative state. Besides, for small drones, visual detection can only be effective at short distances. Detection and relative state estimation may fail when performing cooperative missions with loose formations. Finally, visual features-based methods [20], [21] relies on environmental texture features, which only works in narrow environments. Beyond this, these methods require a consistent heading of the drone swarm.

III. SYSTEM OVERVIEW

Our aerial swarm contains up to n drones. A drone attaches with an id i, will be denoted as drone i. In order to help understand the estimation, the notations are defined in Appendix A and followed by the rest of the paper.

A. Problem Formulation

For a swarm of drones, the state estimation problem can be represent as, for every drone \( k \in \{1,2,3 \ldots n\} \), estimate 6-DOF pose \( v_k^T_1 \) for every drone \( i \in \{1,2,3 \ldots n\} \) at time \( t \) in drone \( k \)'s local frame.

The state estimate problem can be split into two parts,
1) Estimate the ego-motion state of itself in a local frame, i.e. \( \hat{T}^0_k \).
2) For a drone \( k \) estimating the state of any other arbitrary drone \( i, \) i.e. \( v_k^i \) and 4-DOF relative state \( \hat{b}_k^i \).

The VIO utilizes the gravitational acceleration measured by the IMU to help extract the roll and pitch angles in the attitude. Since the consistency of gravity acceleration among drones, with the estimation of the 4-DoF pose \( \hat{v}_k^i \) and relative pose \( b_k^i \), we are able to combine the drone’s own VIO \( \hat{T}^i_k \) to obtain the 6-DoF pose:

\[
v_k^i \hat{T}^i_k = \begin{bmatrix}
 R_z \left((v_k^i \hat{P}^i_1) \psi - (\hat{P}^i_t) \psi \right) & R^t_i & (v_k^i \hat{P}^i_t) X \\
 0 & 1 & 0
\end{bmatrix},
\]

where \( R_z \left((v_k^i \hat{P}^i_1) \psi - (\hat{P}^i_t) \psi \right) \) eliminate the yaw drift of the rotation \( \hat{R}^i_t \) estimated by VIO.

In our previous works, we focused on estimating \( b_k^i \), which is the relative state estimation for the aerial swarm. The ego-motion parts was directly estimated by VIO, \( \hat{T}^i_k = \hat{T}^i_k \) was assumed. However, simply VIO suffers from long-term drift. In this paper, by adopting the map-based relative localization and re-localization method, we are able to directly estimate the state of the aerial swarm without losing the global consistency, which means both \( \hat{T}^i_k \) and \( b_k^i \) are estimated by the proposed state estimator for the aerial swarm.

B. System Architecture

Our proposed method is divided into front-end and back-end. In the front-end, the raw measurements are pre-processed by VIO, map-based localization. In the back-end, graph-based optimization is utilized for state estimation. After that, we utilize the estimated result and the latest VIO to propagate the aerial swarm state.

IV. FRONT-END: MEASUREMENTS PREPROCESSING AND MODELING

A. FishEye Visual Inertial Odometry

In contrast to the pinhole camera with limited FoV in previous work, two fisheye camera with 235 degrees of FoV is utilized in VIO, visual detection and map-based localization, as shown in Fig. 3c.

Based on VINS-Fusion [6] and previous works on fish eye VIO [22], [23], we developed a fisheye visual-inertial system: VINS-Fisheye for Omni-swarm and make it open source. VINS-Fisheye utilizes the raw input from the fisheye cameras and IMU measurements to estimate VIO. As a result of the massive distortion, it is hard to directly apply the existing visual algorithms directly to the raw image data produced by the fisheye camera. As an alternative, we use the methods described in the article [22], [23] to crop regions of the original image and reproject it into five distortion-free images for later algorithms. An example of raw image and processed distortion-free images is shown in Fig. 3b. After reprojecting the raw fisheye images, the VINS-Fisheye performs VIO on these distortion-free images and IMU data to provide real-time local pose and velocity estimation. Moreover, the VINS-Fisheye is capable of generating depth maps from the stereo fisheye camera and will output the processed distortion-free images for subsequent processing. Due to the long term drifting of VIO, instead of directly using the original odometry, we use the 4-DoF relative pose extracted from VIO, which can be modeled as:

\[
z_k^i = (\hat{P}^i_1)^{-1}( \hat{P}^i_t ) = (P^i_1)^{-1}(P^i_t) + n_vio.
\]
Fig. 2: The system architecture of the proposed state estimation framework for aerial swarm. The data from the onboard sensors are processed, then broadcast to all the other drones. The swarm state estimation on each of the onboard computer collects both onboard and broadcast information, including the relative distance from the UWB modules, the VIO, map-based edges, and the detection results, and perform optimization prediction to obtain real-time relative state estimations. The estimation result is sent back to facilitate the matching procedure of detection and tracking meanwhile serve the planning and control. The ground station obtains the drones’ real-time information to monitor the flight status and concurrently send the commands to the drones. All the communications between the devices are through UWB broadcast.

the noise of the relative pose can be assumed as Gaussian, $n_{vio} \sim N(0, \sigma_{vio}^2)$.

The VIO keyframes contain the distortion-free images generated from the fisheye stereo camera, the external parameters of the camera, and the real-time pose estimation is utilized for further processing.

### B. Visual Detection

The distortion-free images extracted from the upper fisheye camera are also utilized to perform drone detection. YOLOv4-tiny\cite{24, 25} is adopted for detecting 2D bounding boxes of the drones on these extracted distortion-free images, which is one of the state-of-art visual detector based on a convolutional neural network. For our visual detection purpose, the network is trained with our custom data for efficiently detecting our drones. YOLOv4-tiny provides detection ability at a frequency up to 5hz on five images extracted from the upper fisheye camera.

The visual detection measurements are divided into the direction corresponding to the center of the detected bounding box and the inverse depth, corresponding to the bounding box’s scale.

Consider on a drone $k$, a virtual distortion-free camera has a rotation $b_k R_v^c$ and a translation $b_k X_v^c$, focal length $f$ and camera projection function $\pi_v(P_v) \rightarrow u_v$, a bounding box of drone $i$ with center coordinate $P_b = [bx^i, by^i]^T$ and width $w$ has been detected. The visual detection measurement is modeled as

$$z_{uk\rightarrow i} = b_k R_v^c \cdot \pi_v^{-1}(P_b) = b_k R_v^c \cdot \pi_v^{-1}(b_k X_v^t - b_k X_v^c) + n_{det}$$

$$z_{s_{uk\rightarrow i}} = \frac{w}{s \cdot f} \frac{1}{\|b_k X_v^t - b_k X_v^c\|_2} + n_{det}$$

where $z_{uk\rightarrow i}$ is the direction of the detected drone in body frame. $z_{s_{uk\rightarrow i}}$ is the normalized width of the bounding box, which is treated as the inverse depth from the drone to the camera. $s$ is the width of the actual drone, the sensor noise is modeled as Gaussian, $n_{det} \sim N(0, \sigma_{det}^2)$ and $n_{det} \sim N(0, \sigma_{det}^2)$.

However, for the flexibility of the aerial swarm, our drone in the aerial swarm is designed to be homogeneous, so we cannot simply associate the detected drone $i$ to its real id. Beyond this, the data association between frames becomes a complicated multiple object tracking problems. The data association problems are seen as part of the state estimation problem. In this paper, the Nearest-Neighbor algorithm (NN) based on estimation results is utilized for visual data association problems. The state estimator can be initialized with the detected map-based edges or UWB VIO fuse result. The follow-up detection will be labeled with the same ID if it has sufficient overlap with one of the visual tracker’s bounding boxes. Otherwise, we will identify it with the Nearest-Neighbour policy with the estimated result.

### C. UWB measurement

The distance measurements from UWB module can be model as:

$$z_{d_{i\rightarrow j}} = \| \omega X_v^t - \omega X_v^t \|_2 + n_d,$$
where $n_d \sim N(0, \sigma_d^2)$ is the Gaussian noise of the distance measurement. In practice, the $\sigma_d^2$ is near 15cm. Consider the design of the drone, the installation length of the antenna relative to IMU can be ignored in our model. We have tested that the introduction of the installation position of the UWB module into estimation causes strong nonlinearity of the estimator, which leads to unstable solutions for the state estimation, but has no effect on the accuracy.

### D. Map-based Localization Module

In contrast, to directly optimize the map, the multi-drone map-based localization module perform relative localization and re-localization of the drone in the swarm by identifying the locations visited by all drones of the aerial swarm. We pack the information of the visited place, including landmarks and visual features, as a keyframe $F_{t}^{i}$, into visual databases as a map.

To be specified, the algorithm for multi-drone map-based localization is shown in Alg. 1 and the procedure is shown in Fig. 4.

1) Visual Database and Multi-drone map-based Localization: When the map-based localization module receives the VIO keyframes, it uses MobileNetVLAD [26], [27] to extract the global features and uses SuperPoint to extract the landmarks and the corresponding descriptor. [28] As shown in Fig. 5b, corresponding between landmarks from upper and lower cameras are established by performing a Brute-Force match of the landmark descriptors. Then the matched landmarks are triangulated for estimating their 3D positions in the local frame. The global feature and landmarks, together with the odometry and extrinsic, are pack to keyframe $F_{t}^{i}$, which will be broadcasted to the entire swarm later.

Suppose a drone $k$ receives a keyframe $F_{t}^{j}$, where if $i = k$, the keyframe is generated by the drone itself. The drone find the most similar keyframe $F_{t}^{0i}$ compared to $F_{t}^{j}$ in the database by function $\text{KF\_QUERY}$. Next, we re-localize the drone $j$ in $F_{t}^{j}$ in the local coordinate frame of $F_{t}^{0i}$ and extract the relative pose, the result is called a map-based edge $L_{i \rightarrow j}^{0 \rightarrow t}$. These map-based edge $L_{i \rightarrow j}^{0 \rightarrow t}$ is again broadcast to the whole aerial swarm, and all drones that receive it fuse them into their own state estimates. In addition, $F_{t}^{j}$ will be added to drone $k$’s database.

In order to store and retrieve the map, we build databases based on Fais [29], which is a vector similarity retrieval database. The keyframes $F_{t}^{j}$, which are indexed by the corresponding global descriptors, are saved in the databases as maps.

On each drone, there are two separate visual databases, a remote database that stores keyframes broadcast from other drones received by the network and a local database that stores generated keyframes from the local drone.

For new keyframes received from the remote drones, we only retrieve keyframes in the local database, and for local keyframes, we retrieve keyframes in both the remote and local databases. Extraction map-based edge for a pair of keyframes from remote drones is avoided for saving computation resources since all the extracted map-based edge are broadcasted to the whole swarm.

2) Relative Pose Extraction: Once the keyframe $F_{t}^{i0}$ is returned from the visual database, we establish the 2D-3D matches from the landmarks of $F_{t}^{i0}$ to landmarks of $F_{t}^{j}$ by using the landmark descriptors with a Brute-Force matcher $\text{BF\_MATCHER}$. Querying of database results and the Brute-Force matcher may cause outliers. To remove the outliers, the map-based edge is verified with two methods:
Fig. 4: This figure shows the structure of multi-drone map-based localization. First, we process keyframes from the VIO with NetVLAD and SuperPoint to generate keyframes containing global and local features. These keyframes are broadcasted to the whole swarm through the wireless ad hoc network. After that, we search for the most similar historical keyframes in the database with the current keyframes generated locally and the current keyframes received from the wireless ad hoc network and add these keyframes to the database after the search is completed. We match features from the returned keyframe and the current keyframe to perform relative Pose extraction. On successful relative pose extraction, the map-based edge containing the relative pose is broadcast to the whole swarm, and either the map-based edge generated locally or the map-based edge received from another drone is sent to swarm state estimation to complete the state estimation.

Fig. 5: (a) The matched landmarks between upper camera and lower camera. (b) The matched landmarks between different keyframes.

- Perspective-n-point(PnP) test within RANSAC [30] between the 2D features of incoming keyframe to the 3D positions of the queried keyframe. The test is performed by function PnP_RANSAC.
- Geometric test performed by G_CHECK. In this test, the consistency of the gravity is checked with the 6D pose extracted by PnP Ransac.

If enough inlier is found the PnP Ransac and the geometric test is passed, the map-based edge \( L_{t_0 \rightarrow t_1}^{i \rightarrow j} \) is considered to be valid. The corresponding relative pose \( z_{L_{t_0 \rightarrow t_1}^{i \rightarrow j}} \), which represent the relative transformation measurement from the keyframe \( F_{t_0}^i \) to keyframe \( F_{t_1}^j \). In addition, \( z_{L_{t_0 \rightarrow t_1}^{i \rightarrow j}} \) can be modeled as:

\[
\begin{align*}
    z_{L_{t_0 \rightarrow t_1}^{i \rightarrow j}} &= (v_i^t)^{-1} (v_j^t) + n_L \\
    &\quad \mathbf{(5)}
\end{align*}
\]

When \( i \neq j \), the map-based edge \( z_{L_{t_0 \rightarrow t_1}^{i \rightarrow j}} \) provide sufficient observability of the relative pose of drone \( i \) and drone \( j \). This makes map-based edges play an essential role in the observability verification and initialization of state estimation. If \( t_0 \neq t_1 \), the map-based edge represents the relative pose of the drones which visit the same place at different time, which plays the role of eliminating accumulated drifting error of VIO.

V. BACK-END: GRAPH-BASED OPTIMIZATION FOR STATE ESTIMATION

A. Graph-based optimization for Swarm State Estimation

A 4-DoF graph-based optimization for the aerial swarm is proposed for state estimation. The present state estimator is individually running on each drone in the aerial swarm. To simplify notification, the following discussion is on drone \( k \), and the aerial swarm contains \( n \) drones, which are numbered from drone 1 to drone \( n \).

Suppose \( SF^n_t \) denotes the swarm keyframe contains of the keyframes \( [F_1^t \ldots F_n^t] \) of \( n \) drones and measurements at time \( t \) for drone \( k \), including the UWB distance measurements, associated detections. The graph contains \( m \) swarm keyframes and corresponding measurement edges:

\[
G = [SF^1_t, SF^2_t, SF^3_t, \ldots, SF^m_t].
\]

The 4-DoF graph-based optimization is constructed from \( G \). For every swarm keyframe \( SF^k_t \) in the graph, it contains up to \( n \) drone keyframes \( F_1^t \ldots F_n^t \). Each keyframe \( F_i^t \) serves as a vertex in the graph-based optimization. The \( v_i^t \) inside the keyframe \( F_i^t \) represent the pose to be estimated of the drone \( i \) at time \( t \) in drone \( k \)'s local frame. Thus we have maximum \( mn \) vertexs in the graph-based optimization. These vertexs are connected by a few types of edges, including odometry edges,
In addition, outlier rejection mechanisms are designed to reduce the impact of outliers from various measurements, which are proposed in Sect. V-C.

B. Optimization Problem

Once we setup the graph, we can solve the state estimation problem by the graph-based optimization. For a drone $k$, the full state vector $X_k$ of the swarm state estimation problem is defined as:

$$
\begin{bmatrix}
\forall k: \hat{P}_0^T, \forall k: \hat{P}_1^T, \ldots, \forall k: \hat{P}_{l-1}^T, \forall j: \hat{P}_0^T, \ldots, \forall j: \hat{P}_{n-1}^T
\end{bmatrix}^T
$$

where $n$ is the number of drones in the swarm system, $m$ is the number of keyframes in the graph.

A bundle adjustment formulation is adopted for the optimization of the states. By minimizing the Mahalanobis norm of the residuals of the measurements, a maximum a posteriori estimation can be obtained. In addition, 6-DoF pose $^{v_k}\hat{T}_t$ is propagated based on latest VIO follow (1).

The optimization is expressed as the following formulation:

$$
\min_{X_k} \left\{ \sum_{(i,l) \in S} \| r_{vo} (z_{i,l}^{o_{PL}}, X_k) \|^2 + \sum_{(i,j) \in U} \rho \left( \| r_d (z_{i,j}^t, X_k) \|^2 \right) + \sum_{(i,j) \in D} \rho \left( \| r_{det_a} (z_{uk-i}, X_k) \|^2 \right) + \sum_{(i,j) \in L} \rho \left( \| r_{det_s} (z_{uk-i}, X_k) \|^2 \right) \right\},
$$

where $U$ is the set of all distance measurements, $D$ is the set of all visual detection measurements, $S$ is the set of all VIO ego-motion measurements. $r_d(z_{i,j}^t, X_k)$ is
the residual of distance measurements; \( r_{det_u}(z_{uk_i}, \lambda_k) \) and \( r_{det_d}(z_{sk_i}, \lambda_k) \) is the residual of visual detection measurements, \((i, j)\) present the pair of successful detection and tracking of drone \( j \) detected by drone \( i \); \( r_{vo}(z_{i}', \lambda_k) \) represents the residual of VIO results, ensuring the local consistency of the drone \( i \)'s trajectory measured in the two drones’ VIO frames; \( r_{C}(z_{i,t_0-t_1}, \lambda_k) \) represents the residual of the map-based edges, which ensuring the global consistency. Since some anomalous measurements may be generated in distance measurements, map-based edge measurements and detection measurements, we introduce Huber norm \( \rho(s) \) \(^{31} \) to reduce the effect of possible outlier measurements.

Consider for the relative motion between two keyframes \( F_i \) and \( F_j \) for drone \( i \). According to the VIO measurement model defined in \(^2\), the residual of the relative motion is defined as:

\[
 r_{vo}(z_{i}', \lambda_k) = (z_{i})^{-1}(u_{i} - v_{i}) .
\]  

As the measurement model for map-based edges defined in \(^5\), the residual for the map-based edge is also a relative pose residual similar to the relative motion case, which is defined as:

\[
 r_{C}(z_{i,t_0-t_1}, \lambda_k) = (z_{i,t_0-t_1})^{-1}(u_{i} - v_{i}) .
\]

Referring to the UWB distance measurement model defined in \(^4\), the residuals of the inter-drone distance can be defined as:

\[
 r_{d}(z_{dij}, \lambda_k) = z_{dij} - ||v_{i} - u_{i}|| .
\]

C. Outlier Rejection

Measurement outliers can be caused by a variety of factors. For example, UWB can yield large errors due to occlusion, altitude differences, or interference from other wireless sources. Detection outliers are typically caused by false detections or mismatches. Map-based edge errors are usually due to mis-matching of feature points. Although we have performed several outlier rejections on the measurements in the front-end, such as the outlier rejection mechanism in VIO itself, the matching of visual detection targets in Sect. \(^{IV-B}\) outlier rejection and geometric checks on map-based localization in Sect. \(^{IV-D1}\) there will still be outliers that pass these tests, so we have to combine the back-end estimation results to check these values again, as shown in Fig. \(^8\).

Fig. 8: A demonstration of the graph-based optimization of two drones. The green rectangles represent swarm frame from \( S_{F_{t_0}} \) to \( S_{F_{t_1}} \). The yellow arrow denotes the multi-drone map-based edges, the blue arrow denotes the VIO measurement, the green arrow denotes the UWB measurements, and the red arrow denotes the visual detection.
Algorithm 1: Multi-drone Map-based localization Algorithm for Drone \( k \)

**Data:** Local Visual Database \( D_l \), Remote Visual Database \( D_r \), Local Drone \( k \)

**Input:** \( F_j^1 \)

1. Function `KF_QUERY (\( F_j^1, D \) )`
2. \( C \leftarrow \emptyset \)
3. foreach \( S_{K_j}^{t_j} \in F_j^1 \) do
4. if \( j = k \) then
5. \( C \leftarrow C \cup \text{KNN_SEARCH}((F_j^1)_f, D_r) \)
6. \( C \leftarrow C \cup \text{KNN_SEARCH}((F_j^1)_f, D_l) \)
7. \( l_{min} \leftarrow +\infty \)
8. \( F \leftarrow \emptyset \)
9. foreach \( K \in C \) do
10. if \( \|\langle K \rangle_f - \langle S_{K_j}^{t_j} \rangle_f \| < \min(\tau_{f1}, l_{min}) \) then
11. \( F \leftarrow \langle K \rangle_f \)
12. \( l_{min} \leftarrow \|\langle K \rangle_f - \langle S_{K_j}^{t_j} \rangle_f \| \)
13. end
14. return \( F \)

15. Function `G_CHECK (\( V_i \tilde{R}_i^{t_i}, \tilde{T}_i^{t_i}, \tilde{R}_j^{t_j} \) )`
16. \( \delta \tilde{R}_{i \rightarrow j}^{t_i} \leftarrow (V_i \tilde{R}_i^{t_i} - \tilde{R}_j^{t_j}) \)
17. \( \tilde{V}_j^{t_j} \leftarrow \tilde{R}_i^{t_i} \delta \tilde{R}_{i \rightarrow j}^{t_i} \)
18. \( \delta \psi \leftarrow (\tilde{R}_j^{t_j})^T \tilde{R}_i^{t_i} \)
19. \( \Delta \tilde{R} \leftarrow (\delta \psi) \tilde{V}_j^{t_j} \tilde{R}_j^{t_j} \)
20. return \( \|\Delta \tilde{R}\| > \tau_{rot} \)

21. Function `LOOP_DETECTION (\( F_j^1 \) )`
22. \( F_j^1 \leftarrow \text{KF_QUERY} (\( F_j^1, D \) ) \)
23. if \( F_j^1 \neq \emptyset \) then
24. \( P_{2d}, P_{3d_j} \leftarrow \text{BF_MATCHER} (F_j^{t_0}, F_j^{t_1}) \)
25. \( \text{inliers}, V_i^{t_0} \tilde{T}_i^{t_0} \leftarrow \text{PNP_RANSAC} (P_{2d}, P_{3d_j}) \)
26. if \( \text{inliers} \geq \tau_{in} \) then
27. then
28. \( \tilde{T}_i^{t_1} \leftarrow (F_j^{t_1})_T \)
29. \( z_{L_a^{t_0 \rightarrow t_1}} \leftarrow \left( V_i^{t_0} \tilde{T}_i^{t_0} - \tilde{T}_i^{t_1} \right) \_P \)
30. return \( z_{L_a^{t_0 \rightarrow t_1}} \)
31. end
32. end
33. end

In addition to this, since UWB has significant measurement error at large height differences, we also decide whether to use the measurement \( z_{d_{i,j}}^{t_1} \) based on the estimated height difference.

**D. Observability**

We have already established the graph-based optimization for swarm state estimation. However, the state of the aerial swarm is not always observable when measurements collected from each drone are not sufficient. Furthermore, our state estimator is decentralized, running on each drone. The measurements shared on the wireless network may drop due to the environment. Measurements and graph-based optimizations for the individual on each drone are not equal. So the qualitative analysis of observability for each drone is significant for the initialization and further prune of the proposed estimator.

For state estimator running on drone \( k \), suppose the measurements for drone \( j \) is received, the observability of the drone \( i \) can be defined as two levels:

1. 3-DOF Observable: The position \( v_k \mathbf{X}_j^i \) is observable.
2. 6-DOF Observable: The 6-DoF pose \( v_k \mathbf{T}_j^i \) is also observable in this case. When the 4-DoF pose \( v_k \mathbf{P}_j^i \) and VIO \( \mathbf{T}_j^i \) are observable, \( v_k \mathbf{T}_j^i \) will also observable follow \( \square \).

Since the estimator combines multiple measurement information, different combinations of measurements yield different observables. Here we analyze 3-DoF observable and 6-DoF observable one by one.

Table \( \square \) shows the observability of typical measurement combinations. The symbol '+' means whether the measurements are available or unavailable does not change the observability. Visual odometry is always assumed available for drones in the aerial swarm. Otherwise, the drone is not able to stable fly at all. Our estimator has the ability to operate in various environments taking full advantage of different sensor inputs. In narrow environments, such as laboratories, warehouses, and forests, distance measurements and visual detection measurements are occasionally prone to interference, but the rich features allow us to take full advantage of the map-based localization for state estimation. Map-based localization is challenging to perform in open environments, such as stadiums, lawns, and farmlands, where there are few features or far from the aerial swarm. We can use the motion information, distance measurement, and visual detection of the aerial swarm for state estimation instead.

**E. Initialization**

When the system is not initialized, each time a new swarm frame \( SF \) or loop \( L_a^{t_0 \rightarrow t_1} \) is received, the estimator checks if the observability condition is met. The system is initialized when all drones of the swarm met at least the 3-DoF observability condition described in Table \( \square \). When a drone is considered to be observable only in 3-DoF, we will temporarily not estimate its heading value. Due to the optimization problem’s nonlinearity \( \square \), the estimator needs suitable initial values for initialization. In practice, we use random values for initialization and solve multiple times, and the one with the smallest penalty will be chosen as the initialization result.

**VI. System Implementation**

**A. Aerial Platform**

To fully demonstrate the algorithm proposed in this paper, a complete swarm system containing the drone, communication
TABLE I: Observability with typical measurements combinations

| Typical Scene                | Motion of k | Motion of i | Distance | Detection k → i | Detection i → k | Map-based edge Detected | Observability k → i |
|-----------------------------|-------------|-------------|----------|----------------|----------------|-------------------------|--------------------|
| Narrow Indoor and Outdoor Environment (Warehouse, Forest) | -           | -           | -        | True           | -              | True                    | 6 DoF              |
| Open Outdoor and Indoor Environment (Stadium, Farmland) | True        | False       | True     | True           | False          | False                   | 3 DoF              |
|                             | True        | False       | -        | True           | False          | True                    | 6 DoF              |
|                             | True        | True        | True     | True           | -              | True                    | 6 DoF              |

Fig. 9: One of the aerial platforms in the swarm system, which is equipped with stereo fisheye cameras, a DJI N3 flight controller, a Nooploop UWB module and a DJI Manifold2-G onboard computer with a Nvidia TX-2 chip.

network, software management, ground user interface, and algorithm is essential. For this purpose, we first designed a custom drone for the aerial swarm. The drone is equipped with an onboard computer with GPU, as well as sensors including a fisheye camera, UWB, and WiFi module for communication. Secondly, a wireless ad hoc network is set up for real-time communication and firmware updating. Finally, we further designed a distributed multi-agent planning protocol to verify our state estimation method’s real-time performance and practical application capabilities.

Our aerial swarm is consists of five identical drones. Fig. 9 shows one of the drones in the swarm. Our custom drone is equipped with a DJI N3 flight controller, two Pointgrey cameras equipped with fisheye lenses, a NoopLoop UWB module, and a DJI Manifold 2-G onboard computer. The 400hz raw data from the IMU of the N3 flight controller and the 20hz images from the stereo fisheye cameras were fused together by VINS-Fisheye to obtain VIO. As shown in Fig. 10 the drone uses VIO for position control. The position controller receives commands from the planning algorithm and generate attitude and thrust command. The N3 flight controller executes the generated attitude and thrust commands. In this process, all computations, including trajectory planning, state estimation, and flight control, are done on the onboard computer. The communication between the drones and the communication between the drone and the ground station is done by the UWB module and wireless ad hoc network.

With the GPU of the onboard computer, we have accelerated various algorithm modules covered in the paper, including the acceleration of the VINS-Fisheye front-end with CUDA and the acceleration of various convolutional neural networks used for state estimation with TensorRT. On TX2, the TensorRT accelerated CNNs are two times faster than the original implementation using TensorFlow and PyTorch and takes up less memory.

To facilitate the distribution of all algorithms, including state estimation and planning, we divided the system running on DJI Manifold 2-G into two layers: the boot and driver layers running on the system base image and the algorithm layer running in the docker image. The docker images are distributed through our wireless ad hoc network, and each drone automatically pulls updates to improve development efficiency.

B. Network Setup

The latency, bandwidth, interference immunity, and distance of the swarm communication directly determine if the proposed method will work well in practice. In the case of the proposed method exchanges image descriptors and flight paths, requiring much larger communication bandwidth, we use a wireless ad hoc network to meet the communication requirements, which coverages requirements for task execution and system maintenance. With the 5.8Ghz wireless ad hoc network, our communication network is able to cover sufficient distance and provide high bandwidth and low latency communication services.

During the execution of the mission, communications including data exchange for status estimation, path exchange involved in obstacle avoidance among drones, telemetry commands sent from the ground station to the drone, and monitoring of real-time status are all implemented wireless ad hoc network. Some of the transmissions with lower bandwidth requirements are simultaneously backed up using UWB broadcasts. We use Lightweight Communications and Marshalling (LCM) [33] in order to achieve efficient data encoding and communication broadcast in the swarm.

System maintenance then requires firmware upgrades and remote login on the communication network along with log data pulling. On the wireless network, we implement automatic upgrades on the docker image. We set up a permanent docker server in the lab as a node in the wireless ad hoc network. This node is accessed through the wireless ad hoc network when the system is automatically started and is automatically updated to contain the latest algorithm image, which is usually very fast thanks to docker’s layer mechanism.
Fig. 10: A demonstration of the planning and control methods for the aerial swarm.

Fig. 11: Our custom 3D User Interface. The drones shown in the figure are real-time estimated by drone 1. The yellow arrows are the real-time detection measurements. The splines are the real-time trajectories generated by each drone for inter-drone collision avoidance.

While not part of the proposed state estimation, this network architecture has accelerated our development and debugging process.

C. 3D User Interface

Fig. 11 shows the 3D User Interface software we developed to efficiently monitor the flight and status estimation status of the aerial swarm and to command the swarm, which can be treat as an alternative to rviz. Our 3D user interface software is based on ROS and WebGL and can run efficiently on a tablet or laptop. In addition to visualizing the aerial swarm’s real-time flight status, the 3D User Interface displays the necessary flight information of each drone, including the trajectory planned by the planner, battery level, remaining endurance time, and control status. Simultaneously, the 3D User Interface allows us to easily give orders to the drones, including takeoff and landing, formation change, and fly to the specified target point. The 3D User Interface has been open-sourced and will provide more features for aerial swarm in the future.

D. Trajectory Planning for Swarm

To demonstrate the capability of our state estimation method, we conduct fully autonomous formation flights and inter-drone collision avoidance tests, as is detailed in Sect. VII-B. In these flights, trajectories of the drones are generated by the method extended from Fast-Planner [1], a real-time trajectory planner based on kinodynamic search and gradient-based optimization. Initially, [1] assumes a static environment and does not consider the interaction with other agents. Therefore, we adapt it to achieve decentralized collision avoidance. Among all drones, trajectories of them are shared through a broadcast network. For each drone, it checks whether any conflict exists between its own trajectory and others’. Whenever a conflict is found, it replans a new trajectory immediately. The replan starts by searching a safe and dynamically feasible initial path, in which motion primitives conflicting with other drones’ trajectories are pruned to avoid an inter-drone collision. The path is further optimized to improve the smoothness, while safety is guaranteed by a collision cost of the swarm [34]. The process is repeated for each drone independently until it reaches the designated goal.

VII. EXPERIMENT AND COMPARISON

To validate the Omni-swarm’s feasibility, accuracy, and practical value, we divide our experiment into two parts: 1) Accuracy comparison using aerial swarm open-loop collection data. 2) Aerial swarm formation flight and inter-drone obstacle avoidance verification.

A. Accuracy Comparison

In the first part of the experiment, the raw data, including images captured by stereo fisheye cameras, IMU measurements, distance measurements from UWB, and ground truth given by the motion capture system, is collected using 2-3 drones flight pre-programmed trajectories as datasets, as shown in the Fig. [12a].

We perform the state estimation of the collected dataset offline on a powerful PC for accuracy comparison. We run several docker containers simultaneously to simulate the onboard computer of each drone in the swarm. These containers use the same docker image, which packages a full set of swarm state estimation packages, including VIO, visual detector, map-based localization, and state estimation.

Fig. [12b,c] shows the estimated trajectories compared to the ground truth and the aligned VIO trajectories on a two drone
Fig. 12: (a) The estimated trajectories of the two drones in an indoor dataset. The black arrows are the map-based edges. The gray arrows are the visual detection edges. Only part of the edges are shown in this figure for better visualization. (b)-(c) The comparison of the ground truth trajectories (blue), the aligned VIO trajectories (orange), and the estimated trajectories (green). The aligned VIO trajectories drift away from the ground truth trajectories while the estimated trajectories closely follow the ground truth.

Fig. 13: These two figures show the estimated trajectories and the VIO trajectories of the two drones in an outdoor dataset. For better visualization of the global consistency, online trajectories (the propagation result) and offline trajectories (the final optimized state in the graph) are shown in the figures. (a) The estimated trajectories of two drones with the map-edges and visual detection edges. The black lines are the map-based edges. The gray lines are the detection edges. Only part of the edges is shown in this figure for better visualization. (b)-(c) The detailed comparison of the estimated trajectories (blue) and the VIO trajectories (orange). After returning to the start point, the estimated trajectories drifts only 0.8% of the trajectories' length while the VIO drifts 2.8%.

dataset. The two trajectories are the real-time state estimated by drone 1. The Table. II and Table. III shows the Absolute Trajectory Error (ATE) of the proposed method to demonstrate the global consistency of the proposed method and relative Error (RE) to demonstrate the relative state estimation accuracy. The UWB edges, the visual detection edges, and the map-based edges are abandoned in different scenarios to verify the effectiveness of the proposed sensor fusion method. Beyond this, the accuracy of VIO trajectories is also shown in the table. In this case, the VIO is aligned with ground truth at the beginning of the flight. Our proposed method achieves centimeter-level accuracy in relative state estimation while ensuring global consistency. When map-based localization is enabled, the ATE is significantly smaller than it is abandoned, proving the proposed method can ensure the global consistency of the estimated state.

In addition to offline accuracy comparison on a powerful PC with datasets, Fig. 14b-d and Table. IV shows the estimated trajectories compared to the ground truth and the aligned VIO of a three-drone inter-drone collision avoidance test. All the state estimation results in this experiment are generated online and in real-time by the onboard computer. The relative state estimation also achieves centimeter-level accuracy in this experiment. We notice that the three-drone inter-drone collision avoidance test’s estimation accuracy is slightly better than the state estimation of the three-drone offline case. The reason is, in the three-drone dataset, the drones are fly parallel with preset trajectories, and relative motion is small in this case.
TABLE II: Accuracy Comparasion on a Two-drone Dataset

| Traj. Lengths of {1,2} | Proposed Method | Without UWB | Without Visual Detection | Without Map-based edges | Aligned VIO |
|------------------------|-----------------|-------------|--------------------------|-------------------------|-------------|
| RE(\(x_{1,2}\)) | 0.060,0.074,0.071 m | 0.075,0.138,0.072 m | 0.126,0.139,0.078 m | 0.109,0.089,0.089 m | 0.209,0.110,0.099 m |
| RE(\(\psi_{1,2}\)) | 1.68° | 1.54° | 2.35° | 3.27° | 1.49° |
| ATE(\(\chi_{1,2}\)) | 0.157 m | 0.134 m | 0.134 m | 0.356 m | 0.604 m |
| ATE(\(\psi_{1,2}\)) | 0.908° | 1.021° | 0.941° | 2.686° | 1.285° |
| ATE(\(\chi_{1,2}\)) | 0.170 m | 0.158 m | 0.175 m | 0.558 m | 0.238 m |
| ATE(\(\psi_{1,2}\)) | 2.029° | 1.565° | 1.611° | 1.888° | 0.759° |

TABLE III: Accuracy Comparasion on a Three-drone Dataset

| Traj. Lengths of {1,2,3} | Proposed Method | Without UWB | Without Visual Detection | Without Map-based edges | Aligned VIO |
|--------------------------|-----------------|-------------|--------------------------|-------------------------|-------------|
| RE(\(x_{1,2,3}\)) | 0.105,0.112,0.070 m | 0.290,0.196,0.087 m | 0.458,1.001,0.922 m | 0.080,0.123,0.072 m | 0.413,0.238,0.117 m |
| RE(\(\psi_{1,2,3}\)) | 2.48° | 1.54° | 2.78° | 3.84° | 1.61° |
| RE(\(\chi_{1,2,3}\)) | 0.079,0.142,0.111 m | 0.128,0.136,0.132 m | 1.052,0.484,0.404 m | 0.406,0.153,0.621 m | 0.279,0.557,0.089 m |
| ATE(\(\chi_{1,2,3}\)) | 1.81° | 1.89° | 3.58° | 3.54° | 7.98° |
| ATE(\(\psi_{1,2,3}\)) | 0.469 m | 0.304 m | 0.588 m | 0.429 m | 0.331 m |
| ATE(\(\psi_{1,2,3}\)) | 3.794° | 3.206° | 2.91° | 6.80° | 5.28° |
| ATE(\(\chi_{1,2,3}\)) | 0.484 m | 0.432 m | 1.296 m | 0.477 m | 0.415 m |
| ATE(\(\psi_{1,2,3}\)) | 3.965° | 4.183° | 11.618° | 5.41° | 4.09° |
| ATE(\(\chi_{1,2,3}\)) | 0.514 m | 0.331 m | 1.101 m | 0.924 m | 0.738 m |
| ATE(\(\psi_{1,2,3}\)) | 3.868° | 3.533° | 3.844° | 6.57° | 3.61° |

Fig. 14: These figures show the estimated trajectories aligned VIO trajectories and ground truth trajectories of a three drone inter-drone avoidance experiment. (a) The estimated trajectories of the drones with the map-edges and visual detection edges. The black lines are the map-based edges. The gray lines are the detection edges. Only part of the edges is shown in this figure for better visualization. (b)-(d) The detailed comparison of the estimated trajectories (blue) and the VIO trajectories (orange). The aligned VIO trajectories drift away from the ground truth trajectories while the estimated trajectories closely follow the ground truth.

TABLE IV: Accuracy Comparasion on a Three-drone Inter-drone Collision Avoidance Test

| Traj. Lengths of {1,2,3} | Proposed Method | Aligned VIO |
|--------------------------|-----------------|-------------|
| RE(\(x_{1,2,3}\)) | 0.077,0.092,0.073 m | 0.120,0.169,0.053 m |
| RE(\(\psi_{1,2,3}\)) | 0.888° | 0.959° |
| RE(\(\chi_{1,2,3}\)) | 0.060,0.066,0.081 | 0.088,0.091,0.032 |
| RE(\(\psi_{1,2,3}\)) | 1.42° | 0.981° |
| ATE(\(\chi_{1,2,3}\)) | 0.089 | 0.154 |
| ATE(\(\psi_{1,2,3}\)) | 0.445° | 0.733° |
| ATE(\(\chi_{1,2,3}\)) | 0.175 | 0.089 |
| ATE(\(\psi_{1,2,3}\)) | 0.701° | 0.275° |
| ATE(\(\chi_{1,2,3}\)) | 0.148 | 0.058 |
| ATE(\(\psi_{1,2,3}\)) | 1.062° | 0.294° |

proves that relative motion is significant to state estimation accuracy.

We also perform an outdoor experiment to verify the practicality, flexibility in plenty of scenarios, and global consistency. Fig. 13a-13c shows the estimation result of a two-drone outdoor dataset. As a result of averaged 235-meter-run in the outdoor environment, the estimated states drift only averaged 1.9m from the start point, which is 0.8% of the trajectory length. As a comparison, the original VIO trajectories drifts averaged 6.35m from the start point, which is 3.3 times compared to our proposed method. One may note through the aligned VIO delivers a good result in this task, which is not practical in real-world scenarios since it requires a motion capture system to be aligned at the very beginning.

captured compared to the inter-drone collision avoidance task, which


**B. Formation Flights and Inter-drone Collision Avoidance**

In the second part of the experiments, each drone performs independent swarm state estimation in real-time and uses the estimated results for obstacle avoidance and planning, as shown in Fig. 10. The drone’s real-time state estimation results are collected for accuracy comparison, as shown in Fig.14. To simplify the operation, we put a few preset formations on the 3D UI, users can randomly choose the formation, and the aerial swarm will transform to the next formation with real-time decentralized inter-drone avoidance. Fig. [14] shows the visualization of real-time telemetry trajectories generated by the inter-drone collision avoidance in 3D UI. In addition to preset formations, users may also use a mouse to drag the aerial swarm or the single drone to an alternative point in the 3D UI, and the drones will fly to the target point with inter-drone obstacle avoidance.

Fig. 14 shows the real-time state estimation result of a three-drone inter-drone collision avoidance experiment. The result successfully verifies the proposed swarm state estimation method’s real-time performance and accuracy on an onboard computer.

**VIII. Conclusion and Future Work**

In this paper, we introduce *Omni-swarm*, a complete aerial swarm system with custom drones, state estimation, inter-drone collision avoidance, communication, and ground user interface. A decentralized Omni-directional visual-inertial-UWB fusion state estimation for the aerial swarm is introduced as the core of the Omni-swarm. The proposed method combines detection, VIO, and distance measurements, map-based localization and provides the estimation result based on the optimization.

Compared to previous works, the proposed work solves the complicated initialization issue, restricted FoV issue, and global consistency issue. The state estimation is tested by extensive aerial swarm flight experiments, including open-loop formation flight and inter-drone collision avoidance test, to demonstrate the feasibility and effectiveness. The state estimation results are compared with ground truth data from the motion capture system, reaching a centimeter-level precision on relative estimation while ensuring global consistency. With the proposed state estimation, formation flights in various complex environments are no longer impossible. Inter-drone collision avoidance is successfully demonstrated based on the proposed state estimation. We believe that Omni-swarm can be widely adopted in different scenarios and on multiple scales.

**Acknowledgement**

We gratefully acknowledge support from Xinjie Yao, Guotao Meng, Au Chin Wai, and HKUST Aerial Robotics Group members.

**References**

[1] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, “Robust and efficient quadrotor trajectory generation for fast autonomous flight,” IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3529–3536, 2019.

[2] K. Wang, F. Gao, and S. Shen, “Real-time scalable dense surfel mapping,” in Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA), 2019.

[3] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K.-T. Cheng, “Act: An autonomous drone cinematography system for action scenes,” in Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA). IEEE, 2018, pp. 7039–7046.

[4] L. Wang, D. Cheng, F. Gao, F. Cai, J. Guo, M. Lin, and S. Shen, “A collaborative aerial-ground robotic system for fast exploration,” in Proc. of the Intl. Symp. on Exp. Robot. (ISER), 2018, pp. 59–71.

[5] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, “Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments,” IEEE Transactions on Robotics, 2020.

[6] T. Qiu, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018.

[7] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1267, 2017.

[8] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, “Openvisn: A research platform for visual-inertial estimation,” in Proc. of the IEEE International Conference on Robotics and Automation, Paris, France, 2020.

[9] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, “Crazyswarm: A large nano-quadcopter swarm,” in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 3299–3304.

[10] A. Jaimison, S. Kota, and J. Gomez, “An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles uav (s),” in 2008 IEEE International Conference on System Engineering. IEEE, 2008, pp. 1–6.

[11] S. Moon, Y. Choi, D. Kim, M. Seung, and H. Gong, “Outdoor swarm flight system based on rtk-gps,” Journal of KISe, vol. 43, no. 12, pp. 1315–1324, 2016.

[12] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, “Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm,” arXiv preprint arXiv:2003.05138, 2020.

[13] V. Walter, N. Staub, A. Franchi, and M. Sasku, “Uvdar system for visual relative localization with application to leader–follower formations of multirotor uavs,” IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2637–2644, 2019.

[14] T. Nguyen, K. Mohta, C. J. Taylor, and V. Kumar, “Vision-based multi-mav localization with anonymous relative measurements using coupled probabilistic data association filter,” arXiv preprint arXiv:1909.08200, 2019.

[15] K. Guo, Z. Qiu, W. Meng, L. Xie, and R. Teo, “Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in gps denied environments,” International Journal of Micro Air Vehicles, vol. 9, no. 3, pp. 169–186, 2017.

[16] K. Guo, X. Li, and L. Xie, “Ultra-wideband and odometry-based cooperative relative localization with application to multi-uav formation control,” IEEE Transactions on Cybernetics, 2019.

[17] T. A. Vidal-Calleja, C. Berger, and S. Lacroix, “Event-driven loop closure in multi-robot mapping,” in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009, pp. 1535–1540.

[18] T. Qiu, P. Li, and S. Shen, “Relocalization, global optimization and map merging for monocular visual-inertial slam,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1197–1204.

[19] A. Ledegerber, M. Hamer, and R. D’Andrea, “A robot self-localization system using one-way ultra-wideband communication,” in Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE, 2015, pp. 3131–3137.

[20] N. Piasco, J. Marzat, and M. Sanfouche, “Collaborative localization and formation flying using distributed stereo-vision,” in Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016, pp. 1202–1207.

[21] M. W. Achtelik, S. Weiss, M. Chli, F. Dellaert, and R. Siegwart, “Collaborative stereo,” in Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011, pp. 2242–2248.

[22] W. Gao and S. Shen, “Dual-fisheye omnidirectional stereo,” in Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS), 2017, pp. 6715–6722.

[23] W. Gao, K. Wang, W. Ding, F. Gao, T. Qin, and S. Shen, “Autonomous aerial robot using dual-fisheye cameras,” Journal of Field Robotics, vol. 37, no. 4, pp. 497–514, 2020.

[24] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,”
\[ \hat{P}_i \] The estimated state.
\[ \delta P_i \] The transformation matrix from time \( t-1 \) to \( t \) of drone \( i \) from the VIO result, i.e. \( P_i^t = \delta P_i^{t-1} P_i^t \).
\[ d_{i,j}^t \] Distance between drone \( i \) and drone \( j \) at time \( t \).
\[ z_{a_{k-i}} \] The measurement of the direction of the detected drone \( i \) in frame for drone \( k \).
\[ z_{v_{k-i}} \] The measurement of the inverse direction of the detected drone \( i \) for drone \( k \).
\[ F_k^t \] The keyframe of the drone \( k \) at time \( t \), which contains the 4D pose \( v_k P_i^t \) for keyframes \( k \) and other essential information of the drone.
\[ cK_k^t \] The keyframe of the drone \( k \)’s virtual camera \( c \) at time \( t \), which contains the global descriptor, local features, virtual camera’s extrinsic and other essential information of the drone. The virtual camera \( c \) is crop from the raw fisheye camera.
\[ SF_k^t \] Equals to \( [F_k^1 F_k^2 \ldots F_k^n] \) The swarm keyframe of the drone \( k \) at time \( t \), which contains of \( n \) keyframes.
\[ G_k \] The graph built on drone \( k \) for state estimation.
\[ (\cdot)_R \] The rotation part of the transformation matrix.
\[ (\cdot)_T \] The corresponding 6-DoF transformation matrix.
\[ (\cdot)_T \] The corresponding 4-DoF transformation matrix.
\[ z_i^t \] The measurement data at time \( t \).