Communication Efficient, Distributed Relative State Estimation in UAV Networks

Shuo Wang, Yongcai Wang, *Member, IEEE*, Xuewei Bai, *Graduate Student Member, IEEE*, and Deying Li

***Abstract*— Distributed estimation of 6-DOF relative states, including three-dimensional relative poses and three-dimensional relative positions, is a key problem in UAV (Unmanned Aerial Vehicle) networks, which generally requires vision-involved iterative state estimation. How to achieve communication efficiency is a crucial challenge considering the large volume of vision data. This paper jointly considers the communication efficiency, latency, and accuracy for distributed relative state estimation involving vision data in UAV networks. The key is to solve a distributed graph optimization problem, which includes two key steps: (1) local graph construction and node state initialization in an initialization phase, and (2) iterative state update and communication with neighbors until convergence in online iteration phase. A communication efficient, Locating Then Informing (LTI) initialization scheme is proposed, which is run only once by each node to initialize each node's local graph and initial states. For online iteration, a RIPPLE-like distributed state iteration scheme is proposed. It inherits the advantages of traditional sequential and parallel methods while avoiding their drawbacks. It enables nodes' states to converge quickly using fewer rounds of communications. The communication costs for the initialization and online iteration processes are analyzed theoretically. Extensive evaluations use synthetic data generated by AirSim (a widely used UAV network simulation platform) and real-world data are presented. The results show that the proposed method provides accuracy comparable to the centralized graph optimization method and significantly outperforms the other distributed methods in terms of accuracy, communication cost, and latency.**

***Index Terms*— Communication efficiency, distributed graph optimization, relative state estimation, UAV networks.**

I. Introduction

THE network of Unmanned Aerial Vehicles(UAVs) has attracted great attention, because of its potential in many applications such as search, rescue, transport, disaster detection, and military. Although the global navigation satellite system (GNSS) is most widely used for UAV localization [1], [2], the GNSS signal is unavailable in indoor or underground environments. Relative state estimation is an alternative method which locates relative locations and poses among UAVs through relative measurements between them without external location signals. Since vision data contains important clues of pose and location, vision-based localization is generally required to calculate the relative poses of UAVs by common view geometry [3], [4] and graph optimization [5], [6]. Relative distances among UAVs can be measured by ranging techniques such as UWB (Ultra Wide Band) [7]. Therefore, estimating 6-DOF relative states among UAVs through onboard vision and distance measurements is an important basis for a variety of applications [8], [9], [10].

The delay and communication cost for transmitting vision data to a backend server is generally unaffordable, therefore, distributed state estimation [11], [12] in which each UAV processes data locally and updates states by distributed calculation is highly required, which is robust against link or node failures. Because each UAV has limited sensing range and cannot obtain faraway information, so distributed iterative optimization methods are widely adopted [13], [14], [15], [16], [17] for information and state propagation in the network. In the iteration process, each UAV uses its local measurements and neighbors' initial states to estimate its own state in a local graph and then sends its updated states to its neighbors. The neighbors repeat this state updating and propagation process until the states of all the UAVs reach convergence. In existing distributed approaches, one category uses only the distances measured by the UWB among UAVs. Then distributed methods like barycentric coordinate based [13], geometry based [14] and distributed graph optimization [15] are used to estimate the 3-DOF position. However, distance-only methods can not recover the 6-DOF relative states. In order to recover the 6-DOF relative states among UAVs, relative poses extracted from common-view images are the main solutions [18], [19].

Graph optimization methods that minimize re-projection errors can be utilized to estimate the relative poses among UAVs through feature point (e.g., ORB features [20]) extraction and matching of common-view features. The feature point extraction [20] and object detection [21] can be conducted efficiently on embedded devices, but vision-based relative pose estimation still needs to solve the iterative graph optimization problem, which needs to consider the internode communication. In the literature, the distributed graph optimization routine has been investigated [16], [17], which contains key steps of distributed initialization of agent states, state propagation and convergence analysis.

Manuscript received 15 April 2022; revised 15 August 2022; accepted 30 November 2022. Date of publication 6 February 2023; date of current version 17 March 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61972404 and Grant 12071478; and in part by the Public Computing Cloud, Renmin University of China. *(Corresponding author: Yongcai Wang.)*

Shuo Wang, Xuewei Bai, and Deying Li are with the School of Information, Renmin University of China, Beijing 100872, China.

Yongcai Wang is with the Metaverse Research Center and the School of Information, Renmin University of China, Beijing 100872, China (e-mail: ycw@ruc.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3242708.

Digital Object Identifier 10.1109/JSAC.2023.3242708

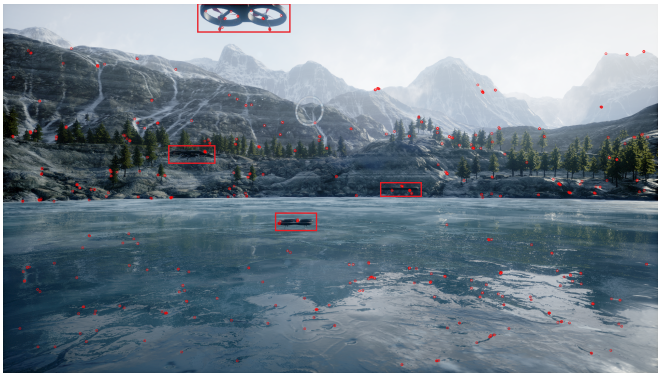

Fig. 1. An image taken by the camera on a UAV. The red boxes are the results of object detection. The red points are the extracted ORB feature points.

But existing studies mainly consider the computation part. How to design a communication efficient scheme to support the distributed computation has been rarely reported, which is a crucial problem especially when vision data is involved. On the one hand, the UAV networks are highly dynamic, which need low-latency state updates. On the other hand, the vision data needs considerable time to be transmitted even after object detection and feature extraction, and the iterative distributed state estimation needs to transmit the updated states repeatedly among nodes before converging. Several existing works [17], [22] adopt an asynchronous iterative scheme to handle communication and state iteration, that all nodes compute and communicate asynchronously. However, existing works have not considered optimizing the communication scheduling to improve the convergence speed and the accuracy of the distributed algorithms. In this paper, we show that efficient communication schemes and distributed graph optimization schemes can be jointly designed for low communication cost, low latency, and accurate vision-involved relative state estimation in UAV networks. The key contributions are as follows:

(1) Using UWB as the inter-UAV distance measurement module and the onboard camera for UAV detection and environment feature point detection, a practical graph optimization model based on multiple feature fusion for UAV relative state estimation is introduced. In particular, a patch-based local graph optimization model (PBM) is proposed, which shows better performance than the traditional node-based local graph optimization model (NBM).

(2) A Locating Then Informing (LTI) scheme is proposed for communication efficient state initialization. In LTI, after the neighbors of the root obtain their initial locations, they send their local measurements to neighbors. Then neighbors can start initialization. Eventually, all nodes obtain their initial states and construct their local optimization graphs, and each node sends data only once during this process.

(3) Then for online distributed state estimation, a RIPPLE-like state update and propagation method is proposed, which takes advantage of both the traditional sequential and parallel state updating schemes while avoiding their drawbacks. It helps the distributed graph optimization to converge both quickly and accurately.

(4) We analyze the communication cost in state initialization and distributed iteration processes. The performances of the proposed PBM + RIPPLE scheme are compared with the centralized method and five other distributed methods by simulations conducted by AirSim [23] (a widely used UAV network simulation platform). The results show that PBM + RIPPLE provides localization accuracy close to the centralized method and outperforms the other distributed optimization methods regarding communication cost, latency, and accuracy.

The remaining sections are organized as follows. Related works will be presented in Section II. The network and perception model are introduced in Section III. A communication efficient initialization scheme is presented in Section IV. RIPPLE-like communication efficient distributed state iteration is introduced in Section V. The evaluation results are presented in Section VI and the paper is concluded with remarks in Section VII.

II. Related Work

In recent years, more and more research has been attracted on how a group of UAVs collaboratively locate and perceive the environment. Distributed relative state estimation is one of the fundamental problems in these studies. Existing algorithms mainly focus on measurement methods and relative state estimation algorithms. We, therefore, introduce related works according to the measurement models and algorithms they proposed.

1) Measurement Models: For UWB-based methods in [24], [25], [26], [27], and [28], they utilize UWB distance measurement to estimate the relative states sometimes with the help of odometry information. However, the estimation results from distance measurement are mainly 3-DOF locations, which cannot infer the 3-DOF poses of the UAVs.

In object-detection-based methods, each UAV uses the onboard camera to observe other UAVs in the field of view. Notably, cameras are inherently bearing sensors. Yang et al. [29] studied the pose state estimation of UAV based on a single image of a typical landing pad which consists of the letter "H" surrounded by a circle. In the system proposed by Saska et al. [30], each UAV is equipped with a marker for detection using pattern recognition. However, marking in the environment or carrying marks on UAVs is not always practical. Yang et al. [18] and Nguyen et al. [19] proposed probability and filtering methods for relative state estimation. However, these methods have limited accuracy. Generally, the method based on object detection will be limited by the FOV of the camera and will fail in some network topologies.

The feature-based methods in [31] and [32] detect the feature points in the environment and restore the relative states between UAVs through the matching of feature points. These methods rely on reliable visual feature point extraction.

Because each sensing technology has its own limitations, recently, the fusion of multiple measurement methods has attracted attention. The system studied by Xu et al. [7] introduces decentralized visual-inertial-UWB fusion for relative state estimation and achieves good accuracy. In [10], Xu et al. proposed an improved algorithm by using more cameras on

the basis of the previous methods, and the environmental landmarks were considered. However, these systems require each UAV to have very high computing power and many pieces of sensor equipment. Moreover, they assume that there is all-to-all communication between UAVs. Therefore, there are restrictions on the scale of the UAV network.

2) Centralized and Decentralized Algorithms: To accomplish the state estimation in UAV networks, existing collaborative localization techniques can be classified into three categories, i.e., centralized, decentralized, and distributed.

In centralized systems, all the data measured by the agents will be sent to and be processed at a central station (or a central UAV) [24], [33], [34]. Common view feature extraction and graph optimization will be conducted at the central server. For range-only based UAV network relative localization, Wang et al. [35] proposed network formation calculation methods using global rigid graph component stitching for pursuing accurate and robust localization performances. Ping et al. [36] investigated conditions and algorithms for accurate localization in sparse networks, which is suitable for UAV networks because the distance measurements in UAV networks are generally sparse. After the central server sends back the estimated results, every agent obtains its states [37]. But long-distance visual data transmission and centralized processing will cause a certain delay, which is unsuitable for the high-speed UAV network. Another issue is that once the central station is damaged or some communication links are blocked, the relative positions of some UAVs will be lost.

In decentralized approaches, the data processing and computation are distributed to several representative agents in the group. Each representative agent manages a group of agents. Data is collected to the representative agent, and the states of members are calculated. The decentralized methods introduced in [7] and [10] are flexible and robust. However, each representative node needs the ability as the central node. The decentralized methods also involve considerable latency for multi-hop visual data collection. Another problem is that the representative nodes consume energy much faster.

3) Distributed Algorithms: Fully distributed approaches have also been investigated in the literature. Although positioning is a nonlinear problem, Safavi [38] discussed a cooperative linear distributed iterative solution with only local measurements, local communication, and local computation needed at each agent. The distributed linear iteration methods are based on barycentric coordinate representations of node locations and are sensitive to measurement noises [13], [39]. Recently, Ping et al. [40] investigates the node localizability problem in barycentric coordinate based distributed localization and shows that the localizable nodes should be detected at first for correct convergence of the distributed algorithm. Meanwhile, the convergence speed is slow, limiting this method's application scope.

Because each UAV has the ability of observation, decision-making, and autonomous movement, the UAV network is regarded as a multi-agent reinforcement learning system in some research work [41]. Peng et al. [42] introduced how to solve the localization problem with deep reinforcement learning in a distributed manner without a central controlling unit. However, the method has a large number of parameters, and their portability has not been proved.

Graph optimization is one of the fundamental techniques for simultaneous localization and mapping (SLAM). It is widely used in the localization of UAV swarms. However, the centralized graph optimization method is unsuitable for large-scale and fast-moving UAVs for the drawback of communication and computing power, while distributed graph optimization performs well. For the distributed perspective, Cheng et al. [15] proposed a distributed algorithm for sensor localization based on the Gauss-Newton method. However, it can only estimate the 2D position by only distance measurement. In DDF-SAM [16] and DDF-SAM2 [12], each agent saves an optimization graph of its surrounding neighbors and optimizes it. Choudhary et al. [43] introduced the Distributed Gauss-Seidel (DGS) approach, which is used in SLAM systems [44], [45]. This method transforms the graph optimization problem into a linear problem and then solves it in a distributed Gauss-Seidel algorithm. Tian et al. [17] proposed the ASAPP (asynchronous and parallel distributed pose graph optimization) method based on distributed gradient descent. It takes communication into account and solves the problem in an asynchronous way. Chang et al. proposed Kimera-Multi [46] for multi-robot SLAM utilizing ASAPP.

4) Considerations on Communication: Although distributed graph optimization needs iterative state updates and repeatedly local communication to solve, few works have considered the communication aspect. ASAPP in paper [17] designs a communication mode based on the Poisson clock in the communication part to support the distributed optimization. When a Poisson clock cycle ends, the node communicates with its neighbors and sends the latest updated results. In essence, the method is still the parallel iteration in divided time. The method proposed by Bedi et al. [22] considers an asynchronous processing architecture with delays introduced by each node to ameliorate the computational bottleneck associated with synchronized computation and the communication rounds among the nodes. But existing work has not considered optimizing the communication scheduling scheme to speed up the convergence of the distributed optimization.

Overall, related works focused either on the measurement models for prototype system development or focusing on the distributed iterative algorithm design. However, in vision-involved distributed graph optimization, the data communication cost and the iteration latency must be considered. Joint optimization of the communication delay, communication cost, and estimation accuracy is required.

III. Network and Perception Modeling

A. Network Model

We consider an aerial network containing N UAVs, which are assumed to be flying in areas with weak satellite signals or in GNSS-denied environments. Each UAV has two sensors onboard: (1) the UWB ranging device, which measures distances to other UAVs within its ranging scope, and (2) a camera that captures images. UAV fuses these two kinds of

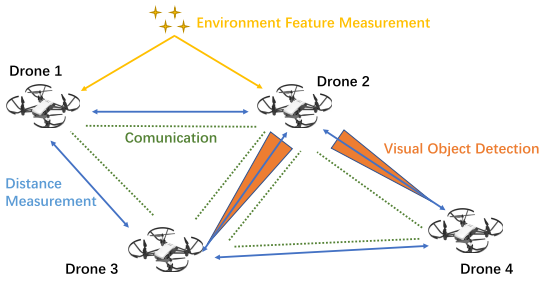

Fig. 2. Here is one example of the UAV network. The distance edges can be built when two UAVs are within a certain distance. For the vision part, UAV 2 is detected by UAV 3 and UAV 4 so that they have the object detection edges. The environment feature edge between UAV 1 and 2 can be built as they can see enough of the same field of view.

data for estimating relative 6-DOF states to other UAVs. Each UAV also equips a communication module. We assume the communication radius is not less than the ranging radius of UWB. So there are two underlying graphs in the UAV network: (1) the communication graph and (2) the observation graph.

The communication graph is assumed to be a simple Unit Disk Graph (UDG). Two UAVs can communicate with each other when they are within communication scope. The observation graph is composed of different kinds of links. A "distance link" exists between two UAVs if they are within the UWB's ranging scope. The vision data constitutes two kinds of links. The first kind, which is constructed by UAV detection [21] from the captured images, is called the "object detection link". The second kind is constructed by common views to environment features, which is called "common view link". A detailed introduction of these links will be given in the perception model. Fig. 2 shows an example of a UAV network and its related communication and observation graphs. The green dashed lines show the communication links. The solid blue lines are the distance links. The brown cones indicate the object detection links, and the yellow links are the links formed by common views of environment features.

For relative state estimation, we assume the UAVs are numbered from 1 to N. The body frame of UAV i is defined as ${}^{b}\boldsymbol{X_i}$. The global frame is defined by the real-time body frame of UAV 1 without loss of generality. ${}^{w}\boldsymbol{X_i}$ describes the state of UAV i in the global frame. We denote the transformation from the body frame of UAV i to its global frame by $\boldsymbol{T^i_{wb}}$, which includes the rotation $({}^{w}\boldsymbol{X_i})_{\boldsymbol{R}} \in SO(3)$ and the translation vector $({}^{w}\boldsymbol{X_i})_{\boldsymbol{t}} \in \mathbb{R}^3$. The relative state estimation problem is to estimate the realtime states of UAVs, i.e., ${}^{w}\boldsymbol{X} = \{{}^{w}\boldsymbol{X_1}, {}^{w}\boldsymbol{X_2}, \ldots, {}^{w}\boldsymbol{X_N}\}$.

Note that this paper focuses on the communication and computation scheduling in the application layer. It assumes there is a proper MAC protocol to handle the collision avoidance problem in data transmission. Recent work in MAC layer [47], [48] can be explored to support the time-division neighborhood data transmission. The UAVs are also assumed to be synchronized via wireless time synchronization protocol [49], [50]. The time delays for communication and collision avoidance are also carefully counted in the communication time analysis and are simulated in a time-division manner in the simulations.

B. Perception Model

1) Distance Function Model: For UWB measurements, each UAV measures the distances to the neighboring UAVs. Let $\hat{d}_{ij}$ be the measured distance between UAV i and j if they are in the measurement threshold D_{uwb}. The value $\hat{d}_{ij}$ is modeled with a Gaussian noise:

$$\hat{d}_{ij} = \|({}^{w}\boldsymbol{X_i})_{\boldsymbol{t}} - ({}^{w}\boldsymbol{X_j})_{\boldsymbol{t}}\| + N(0, \sigma_d^2) \tag{1}$$

where $({}^{w}\boldsymbol{X_i})_{\boldsymbol{t}}$ means the translation part of the state, σ_d is the standard deviation of distance measurement noise, and the $\|\cdot\|$ denotes the second norm of the vector.

2) Reprojection of Detected UAVs: For the camera on each UAV, let us denote the focal length is f and the image size is $L \times W$. The center point coordinate in the pixel coordinate system is (c_x, c_y). The camera intrinsics matrix $\boldsymbol{K}$ can be obtained from the above parameters. The camera frame is defined, and the transformation from the camera frame to the body frame is $\boldsymbol{T^i_{bc}}$ on the ith UAV, which is offline calibrated. The images extracted from the camera are utilized to perform object detection and feature points extraction.

By onboard target detection module YOLO-lite [21], a UAV can detect other UAVs from its captured image if other UAVs appear in the image. The UAV detector will return the 2-D bounding box of the UAVs in the image plane. We consider that by image-based ID embedding [51], the detector can return the ID of the detected UAV. For example, if the UAV i detects the UAV j from its captured image, we can get a 2-D bounding box of UAV j from the image captured by i. The center pixel point of the bounding box is denoted by $\boldsymbol{u_j}$.

According to the pin-hole camera model [52], we can formulate the reprojection coordinate $\hat{\boldsymbol{u}}_{\boldsymbol{j}}$ on UAV i's image plane:

$$\hat{\boldsymbol{u}}_{\boldsymbol{j}} = \frac{1}{s_j}\boldsymbol{K}\ {}^{c}\boldsymbol{Y_{i,j}} = \frac{1}{s_j}\boldsymbol{K}(\boldsymbol{T^i_{wb}T^i_{bc}})^{\boldsymbol{T}}\ {}^{w}\boldsymbol{X_j} \tag{2}$$

where s_j is the depth of UAV j relative to UAV i's camera center, and ${}^{c}\boldsymbol{Y_{i,j}}$ means the position of UAV j in the camera frame of UAV i. Note that ${}^{c}\boldsymbol{Y_{i,j}} = (\boldsymbol{T^i_{wb}T^i_{bc}})^{\boldsymbol{T}}\ {}^{w}\boldsymbol{X_j}$, where $\boldsymbol{T^i_{wb}}$ is the transformation matrix from body frame to world frame on UAV i; $\boldsymbol{T^i_{bc}}$ is the transformation matrix from camera frame to body frame on UAV i; and ${}^{w}\boldsymbol{X_j}$ is the world frame coordinates of UAV j.

3) Model by Common View Feature Points: We extract the ORB feature points [20] from the images taken by the camera. If two adjacent UAVs have enough overlapping fields of view, they can observe some common environmental feature points. The relative state can be recovered by performing a brute-force matching of the feature descriptors.

It is assumed that UAV i and j observe two groups of M corresponding feature points, which are $\boldsymbol{z_i} = \{\boldsymbol{z_i^1}, \boldsymbol{z_i^2}, \ldots, \boldsymbol{z_i^M}\}$ and $\boldsymbol{z_j} = \{\boldsymbol{z_j^1}, \boldsymbol{z_j^2}, \ldots, \boldsymbol{z_j^M}\}$. For the sake of argument, we discuss in the local coordinate system of UAV i. The 3-D positions of these M feature points are defined as $\boldsymbol{P} = \{\boldsymbol{P^1}, \boldsymbol{P^2}, \ldots, \boldsymbol{P^M}\}$. By the projection relationship, the reprojection errors of all the feature points can be obtained. The objective function is to optimize the relative rotation and

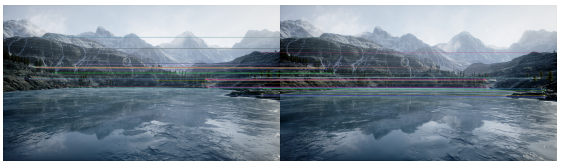

Fig. 3. Feature point matching relationship in the perspective of two UAVs. The ORB feature points are detected. The feature points in the two images are connected, indicating a matching relationship. The RANSAC algorithm [53] is used to eliminate the wrong matching for higher accuracy.

translation, i.e., $\hat{\boldsymbol{R}}_{\boldsymbol{ij}}^{\boldsymbol{c}}, \hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}}$ to minimize the reprojection error:

$$\hat{\boldsymbol{R}}_{\boldsymbol{ij}}^{\boldsymbol{c}}, \hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}} = \min_{\boldsymbol{P}, \hat{\boldsymbol{R}}_{\boldsymbol{ij}}^{\boldsymbol{c}}, \hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}}} \sum_{k=1}^{M} \left\| \frac{1}{s_i^k} \boldsymbol{K} \boldsymbol{P}^{\boldsymbol{k}} - [\boldsymbol{z}_{\boldsymbol{i}}^{\boldsymbol{k}}, \boldsymbol{1}]^{\boldsymbol{T}} \right\|^2 + \left\| \frac{1}{s_j^k} \boldsymbol{K} (\hat{\boldsymbol{R}}_{\boldsymbol{ij}}^{\boldsymbol{c}} \boldsymbol{P}^{\boldsymbol{k}} + \hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}}) - [\boldsymbol{z}_{\boldsymbol{j}}^{\boldsymbol{k}}, \boldsymbol{1}]^{\boldsymbol{T}} \right\|^2 \quad (3)$$

where $\boldsymbol{K}$ is the intrinsics matrix of the cameras, $\hat{\boldsymbol{R}}_{\boldsymbol{ij}}^{\boldsymbol{c}}$ and $\hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}}$ are the relative rotation and translation between two UAVs' camera frames, and s_i^k and s_j^k are the depth of feature point k to the UAV i and to the UAV j.

Equation (3) is a Least-Squares problem that can be solved by Levenberg-Marquardt(LM) algorithm. As a result, we get the 3-DOF position of feature points in UAV i's camera frame and the relative pose $\hat{\boldsymbol{R}}_{\boldsymbol{ij}}^{\boldsymbol{c}}$ and $\hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}}$ between two UAVs' camera frame. Since the camera's extrinsics relative to the body frame have been calibrated, the relative state $\hat{\boldsymbol{R}}_{\boldsymbol{ij}}$ and $\hat{\boldsymbol{t}}_{\boldsymbol{ij}}$ between the body frames of two UAVs can then be obtained.

Using the above network and observation models, we assume N UAVs fly in a GNSS-denied environment. Without loss of generality, we use UAV 1's body frame as the world frame. Other UAVs want to know their relative poses to UAV 1, i.e., poses and positions in the world frame defined by UAV 1.

IV. Communication Efficient Initialization

Since UAVs are flying dynamically, the relative states among UAVs are updated with short cycles. Before conducting any periodical state update, there is an initialization phase. During the initialization, each UAV discovers neighbors and exchanges measurements to set up its local graph model and calculates its initial state, which will be used as the initial state for the first period of distributed iteration. Note that the initialization needs to be conducted only once. In the later periods, the states of the last period will be used as the initial states of this period. The newly captured measurement data will be transmitted along with the state updating message, so the local graph model can be updated accordingly in the distributed iteration process.

The overall routine, including initialization and online distributed iteration, is shown in Fig. 4. When a UAV is not initialized, it enters an initialization phase, which constructs its local graph and obtains its initial states. A Locating Then Informing (LTI) scheme is proposed for communication efficient initialization (IV-B). If a UAV has already been initialized, it will enter the iterative relative state estimation phase, which will be introduced in Section V-B.

This section introduces the initialization phase. We firstly introduce two kinds of local graph models, i.e., node-based and patch-based local graph models (NBM and PBM).

A. Local Graph Model

Each UAV sets up a local graph model for local state estimation based on its measurements to neighbors and the measurement information received from neighbors.

1) The Vertices: In local graph model [54], vertices represent state variables to solve, and edges represent constrains among the state variables. Local graph optimization essentially solves the least square problem of the variables represented by the vertices under the constraints of edges. Considering the local graph model of UAV i, we firstly introduce the types of nodes and edges. Then a node-based local graph model (NBM) and a patch-based local graph model (PBM) are introduced.

In both the NBM and PBM models, the state variables (i.e. vertices) are ${}^{\boldsymbol{w}}\boldsymbol{X}_{\boldsymbol{i}}$ and $\{{}^{\boldsymbol{w}}\boldsymbol{X}_{\boldsymbol{j}}, j \in N(i)\}$, where j is one of i's direct neighbor. These states form the vertices of the local graph model of the UAV i, which is denoted by $\boldsymbol{G}_{\boldsymbol{i}}$. Residue functions between the states set up the edges in the local graph model $\boldsymbol{G}_{\boldsymbol{i}}$.

2) The Edges: Regarding different kinds of measurements, there are four kinds of edges in the local graph model. The distance measurements from i to neighbors contribute the ranging residual edges, where $\hat{d}_{ij}$ is the measured distance and $\|({}^{\boldsymbol{w}}\boldsymbol{X}_{\boldsymbol{i}})_{\boldsymbol{t}} - ({}^{\boldsymbol{w}}\boldsymbol{X}_{\boldsymbol{j}})_{\boldsymbol{t}}\|$ is the distance predicted by states.

$$R_{uwb}(i,j) = w_1 \left(\left\| ({}^{\boldsymbol{w}}\boldsymbol{X}_i)_{\boldsymbol{t}} - ({}^{\boldsymbol{w}}\boldsymbol{X}_j)_{\boldsymbol{t}} \right\| - \hat{d}_{i,j} \right)^2 \quad (4)$$

$R_{uwb}(i,j)$ represents the residue error constructed from the distance measurement from i and one of its neighbor node j; w_1 is the weight.

Also, the object detection and reprojection constraint is formulated from Equation (2):

$$R_{obj}(i,j) = w_2 (\|\boldsymbol{u}_{ij} - \hat{\boldsymbol{u}}_{ij}\|_F)^2 \quad (5)$$

where $R_{obj}(i,j)$ represents the error from reprojection UAV j in UAV i's camera frame. $\hat{\boldsymbol{u}}_{i,j}$ is the measured pixel coordinates of UAV j's center, and $\boldsymbol{u}_{i,j}$ is the predicted pixel coordinates of UAV j's center. $\|\cdot\|_F$ means Frobenius norm and w_2 is the weight.

From matching of the feature points, we get the relative state between UAV i and j: $\hat{\boldsymbol{R}}_{\boldsymbol{ij}}$ and $\hat{\boldsymbol{t}}_{\boldsymbol{ij}}$. We can construct two kinds of constrain edges from them. The rotation error can be defined as:

$$R_{fR}(i,j) = w_3 \left(\left\| \hat{\boldsymbol{R}}_{ij}^{-1} ({}^{\boldsymbol{w}}\boldsymbol{X}_i)_{\boldsymbol{R}}^{-1} ({}^{\boldsymbol{w}}\boldsymbol{X}_j)_{\boldsymbol{R}} - \boldsymbol{I} \right\|_F \right)^2 \quad (6)$$

where $\hat{\boldsymbol{R}}_{ij}$ is the measured relative rotation and $({}^{\boldsymbol{w}}\boldsymbol{X}_i)_{\boldsymbol{R}}^{-1} ({}^{\boldsymbol{w}}\boldsymbol{X}_j)_{\boldsymbol{R}}$ is the predicted relative rotation. Note that $({}^{\boldsymbol{w}}\boldsymbol{X}_i)_{\boldsymbol{R}}$ is the rotation matrix converted from the UAV i' pose in its state ${}^{\boldsymbol{w}}\boldsymbol{X}_{\boldsymbol{i}}$.

However, this translation vector has no scale, but we can restore the scale through the existing ranging information. The translation error is:

$$R_{fT}(i,j) = w_4 \left(\left\| \hat{d}_{ij} \hat{\boldsymbol{t}}_{\boldsymbol{ij}}^{\boldsymbol{c}} - (({}^{\boldsymbol{w}}\boldsymbol{X}_i)_{\boldsymbol{t}} - ({}^{\boldsymbol{w}}\boldsymbol{X}_j)_{\boldsymbol{t}}) \right\|_F \right)^2 \quad (7)$$

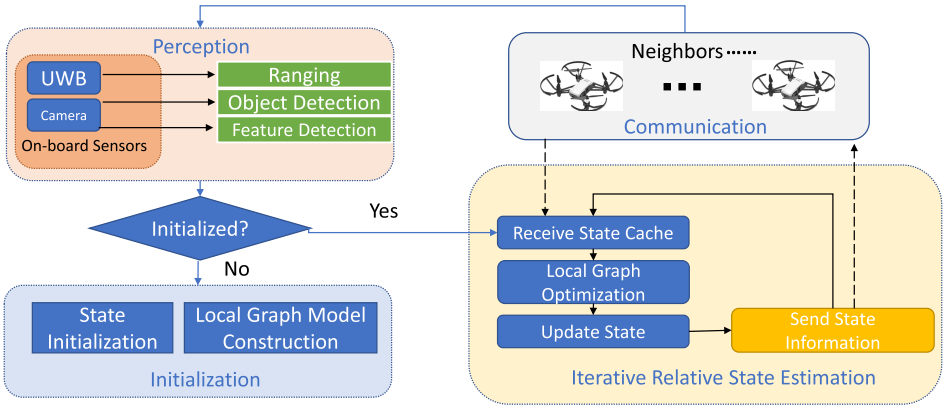

Fig. 4. The main pipeline for one UAV. Each UAV collects the data from the sensors onboard. If it is not initialized, it enters an initialization phase (IV-B). If it has already been initialized, the UAV collects the states of neighbors and performs local graph optimization. The results are utilized to update its own state and broadcast to neighbors. Through iteration, the error is continuously reduced, and an accurate result is obtained.

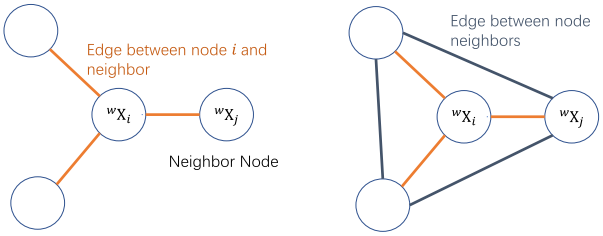

Fig. 5. Node based model: NBM (left) and patch based model: PBM (right).

where $({}^{w}\boldsymbol{X}_i)_t - ({}^{w}\boldsymbol{X}_j)_t$ is the predicted relative position. $({}^{w}\boldsymbol{X}_i)_t$ is the position part in the state ${}^{w}\boldsymbol{X}_i$.

3) Node-Based Model (NBM): In the past, localization methods based on graph optimization mainly adopt node-based local graph model [7], [10], [43], [55]. In the node-based local graph model, only the edges between i and its direct neighbors, i.e. $N(i)$, are considered in the local graph of UAV i. Let $\mathbf{R}(i,j) = R_{uwb}(i,j) + R_{obj}(i,j) + R_{fR}(i,j) + R_{fT}(i,j)$. So the sum of residues at UAV i is:

$$R({}^{w}\boldsymbol{X}_i) = \sum_{j \in N(i)} \mathbf{R}(i,j) \tag{8}$$

However, we find that such a local model does not make full use of the information between neighbors. Fewer edge constraints will reduce the accuracy of local state estimation. Therefore, we propose a patch-based model to improve the utilization of local information.

4) Patch-Based Model (PBM): In patch-based model, not only the edges between UAV i and its direct neighbors, but also the edges among its direct neighbors are considered at UAV i. So the sum of residues at UAV i is:

$$R({}^{w}\boldsymbol{X}_i) = \sum_{j \in N(i)} \mathbf{R}(i,j) + \sum_{j \in N(i)} \sum_{k \in N(i), k \neq i,j} \mathbf{R}(j,k) \tag{9}$$

where $N(i)$ is the neighbor set of UAV i. Note that i considers the residues with direct neighbors and the residues among the direct neighbors. The difference between NBM and PBM for the UAV i can be visually seen from Fig. 5.

At UAV i, Equation (8) and (9) can be solved by Levenberg-Marquardt(LM) algorithm distributively, where the trust region with sparse normal Cholesky method can be chosen [54].

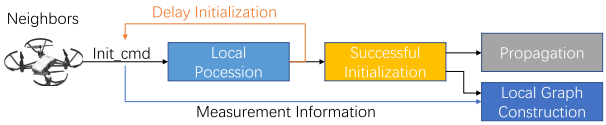

Fig. 6. The pipeline of initialization for uninitialized UAVs.

B. Initialization Phase

One difficulty of initialization is that all UAVs have no global states, except for UAV 1, designated as the global frame's origin. The key to initialization is to use local information and local communication to make UAVs obtain the initial states relative to UAV 1 in the shortest possible time.

We propose a Locating Then Informing (LTI) scheme for communication-efficient initialization. For clarity, we introduce the initialization process in PBM. The initialization process in NBM is analogous.

1) Initialization in PBM: At first, UAV 1 broadcasts its state. The format of the message is defined by the Initialization Command and the content is:

$$[Init\_cmd, ID = 1, {}^{w}\boldsymbol{X}_1, \{\hat{d}_1\}, \{\hat{\boldsymbol{u}}_1\}, \{\hat{\boldsymbol{z}}_1\}]$$

where $\{\hat{d}_1\}$, $\{\hat{\boldsymbol{u}}_1\}$, and $\{\hat{\boldsymbol{z}}_1\}$ represent the set of ranging, object detection, and feature point extraction measurements respectively. Once one UAV receives the *Init_cmd* message from its neighbors(e.g. UAV 1), it starts to execute the initialization algorithm. For the sake of argument, we take UAV i as an example to introduce the algorithm as shown in Fig. 6:

1) **Local processing.** After UAV i receives the *Init_cmd* from UAV j, it will process the measurement data. Especially, it matches its local feature point sets $\{\hat{\boldsymbol{z}}_i\}$ with $\{\hat{\boldsymbol{z}}_j\}$. If enough matching points are found (enough common view features), the relative state $\hat{\boldsymbol{R}}_{ij}$ and $\hat{\boldsymbol{t}}_{ij}$ between the body frames of two UAVs will be determined. At this time, according to the initialized state of j: $({}^{w}\boldsymbol{X}_j)$, UAV i will initialize its state.

$$ {}^{w}\boldsymbol{X}_i = \hat{\boldsymbol{R}}_{ij} \cdot {}^{w}\boldsymbol{X}_j + \hat{d}_{ij}\hat{\boldsymbol{t}}_{ij} \tag{10}$$

2) **Propagation.** If UAV i completes the state initialization, it will sends an initialization command to its neighbors.

The message content is

$$[Init\_cmd, ID = i, {}^{w}\boldsymbol{X_i}, \{\hat{d}_i\}, \{\hat{\boldsymbol{u}}_i\}, \{\hat{\boldsymbol{z}}_i\}]$$

3) **Successful initialization or delayed initialization.** If there is distance measurement and at least one kind of visual observation information between UAV i and its initialized neighbors, UAV i can be successfully initialized. If it is initialized, UAV i will broadcast an initialization command to its neighbors. If the initialization fails, UAV i will go into delayed initialization state until a new message that can help it finish the initialization.
4) **Local Measurement Graph Construction.** After the node i initializes its states, and obtains measurements from its direct neighbors, it can set up its local measurement graph using Equation (4)-(7) and (9).

Note that in the measurement set $\{\hat{d}_i\}, \{\hat{\boldsymbol{u}}_i\}, \{\hat{\boldsymbol{z}}_i\}$ broadcasted from a UAV i, it contains the measurements to all its direct neighbors. So when a node i receives measurements from a neighbor j, it also receives the measurements from node j to j's neighbors, denoted by $N(j)$. So after receiving initialization messages from the neighbor set $N(i)$, without additional communication cost, i has known the measurements among its neighbors' neighbors. Therefore, in PBM, after relative state initialization, the only difference with NBM is the local graph construction step.

In PBM, after a node i receives initialization commands from its neighbors, it will setup a neighbor list $N(i)$ and extract its neighbor's measurements to its direct neighbors, i.e., measurements among $N(i)$. Based on these information, the PBM local graph model is constructed.

Remark: we can see that NBM and PBM have the same communication process and the same communication cost in the initialization part. The only difference is the local measurement graph construction, in which PBM utilizes more measurement information. Later we will see that the denser local graph constructed by PBM can obtain more accurate state estimation performance in the experiments.

C. Analysis of Communication Cost in Initialization

In initialization, either in NBM or in PBM, each UAV needs only to send the initialization command once to its neighbors. The distances, state, ID number, and message type in the initial command message occupy constant bytes. The amount of data sent in the initial command is determined by the number of perceived neighbors $N(i)$ and the number of collected environment feature points N_c. Let Δ be the maximum node degree. The communication cost in the initialization phase at each node is $O(\Delta + N_c)$.

During the initialization process, N nodes need to send the initialization message, so the total communication cost of the system initialization is $O\left(N(\Delta + N_c)\right)$.

V. Distributed Graph Optimization

In distributed graph optimization, each UAV solves a local graph optimization problem to estimate local states; broadcasts and receives local states to (from) neighbors; then updates local states and continues above state updating and negotiation steps iteratively until convergence.

There are two critical parts in distributed graph optimization: local graph optimization and iteration scheme. The local graph optimization algorithm is how each node uses the existing local information to update its current state. The iterative scheme is how the local states are propagated. We will introduce two existing naive iteration schemes and propose a RIPPLE iteration scheme in V-B and present the analysis in V-C.

A. Local Graph Optimization

In the PBM model, when one UAV constructs the local measurement graph, a graph optimization algorithm such as G2O [54] can be applied to update the UAV's state to best match the local measurements. Then the UAV will broadcast its updated states to neighbors and receives neighbors' states to form a new objective function. The series of steps is called one round of iteration for a UAV.

The loop of local state optimization, state transmission, and the objective function updating will continue until convergence, i.e., when the local states before and after local graph optimization have a slight difference.

The iterative distributed graph optimization process desires fewer rounds of iteration and quick state updating in each round for low latency and communication efficiency. The PBM model can generally converge in much fewer rounds than the NBM model. So we will introduce the iteration scheme in the PBM model only and compare the performances with NBM in the evaluation section.

B. Iteration Schemes

The iteration process is the key factor affecting the time duration of state updates before convergence. Traditionally, there are two major iteration schemes [15]: sequential iteration and parallel iteration.

Let ${}^{w}\boldsymbol{X}_i^{(p)}$ denote the state of UAV i after the pth iteration has finished. Node i is denoted as $i^{(p)}$. Notably, each UAV only needs to transmit its updated state information to neighbors during the iterations without transmitting local measurement information. This is because the states can converge quickly before the measurements change much and is also for avoiding the cost of frequent measurement data transmission.

1) Sequential Iteration Scheme: For the sequential iteration scheme, the UAV executes the local graph optimization algorithm in sequence according to the UAV ID [15]. The node with the smaller ID calculates local states by local graph optimization at first and then sends its state to its neighbors. Only after node i updates its state will the node $i+1$ update its state by local graph optimization using the updated neighbor states.

When UAV i wants to update in round $p+1$, $\{{}^{w}\boldsymbol{X}_j^{(p+1)}, j \in N(i), j < i\}$ is available. For the rest of neighbors with ID larger than i, the states that are still in the last iteration, i.e., $\{{}^{w}\boldsymbol{X}_j^{(p)}, j \in N(i), j > i\}$ are available. So i uses the partially updated states in its neighborhood to

update the error function. The error function has three parts:

$$R(i) = \sum_{j,k\in N(i),j<i,k<i} R(j^{(p+1)},k^{(p+1)}) + \sum_{j,k\in N(i),j<i\le k} R(j^{(p+1)},k^{(p)}) + \sum_{j,k\in N(i),i\le j,i\le k} R(j^{(p)},k^{(p)}) \quad (11)$$

This equation is equivalent to Equation (8), but the difference is that the iteration flag, i.e., (p) and $(p+1)$ are added to indicate the states of the nodes. By solving it, we get ${}^{w}\boldsymbol{X}_{i}^{(p+1)}$ and node i broadcasts its updated state. Then node $i+1$ updates its state sequentially.

2) Parallel Iteration Scheme: For the parallel iteration scheme, in round $p+1$, all UAVs will not wait for the updated states of neighbors. They use neighbors' states from the results in the last iteration and update their states in parallel. So for all UAVs, the objective function in round $p+1$ is built by $\{{}^{w}\boldsymbol{X}_{j}^{(p)}, j \in N(i)\}$. The object function is defined as:

$$R(i) = \sum_{j,k\in N(i)} R(j^{(p)},k^{(p)}) \quad (12)$$

All UAVs minimize this cost function to get the values of ${}^{w}\boldsymbol{X}_{i}^{(p+1)}$ and then update the states in round $p+1$ in parallel.

These two methods both have their advantages and drawbacks. In sequential iteration, each UAV uses the updated states of neighbors with ID less than it in each round, which generally needs fewer rounds to converge. However, all UAVs update state in order, so it takes longer time to complete a round of iteration. In particular, when the number of UAVs is large, the system cannot ensure real-time performance. The UAVs can update states in parallel in the parallel method, so one round iteration can be finished more quickly. Nevertheless, the UAV cannot utilize the new states of neighbors in the current round, resulting in that the network states generally need more rounds to converge. The transmission of the updated states in the current round also takes considerable time.

3) RIPPLE Iteration Scheme: To overcome the problems in sequential and parallel iteration schemes, we propose RIPPLE, an iteration scheme with the advantages of sequential and parallel iteration but can avoid their limitations.

In RIPPLE, nodes can use the updated states of some neighbors, and some nodes can execute the algorithm in parallel. It is called RIPPLE because the order of state update in the network is similar to the water ripple. We take node 1 as the starting point of the iteration. Neighbors in the same hop are defined in the same layer. From a global perspective, all nodes will be divided into different layers (Fig. 8). In other words, node 1 is taken as the root node to form a width-first spanning tree. The layer of the spanning tree is equivalent to the layer of the iteration algorithm.

When a node completes this round of iteration, it will communicate with its neighbor nodes to send the updated states. In the first round of iteration, node 1 (layer 1) starts the iteration first and sends the state information to the direct neighbors, i.e., the nodes of layer 2. Then nodes in layer 2 update their states in parallel and inform their updated states to nodes in layer 1 and layer 3. Then nodes in layers 1 and 3 will update their states in parallel and inform their updated states to neighbors. Then nodes in layer 2 and layer 4 will update in parallel and the process will continue.

After all nodes have completed the first round of iteration, we can discuss the iteration order by parity. We denote that lay_L is a set of nodes in layer L. When the node i from layer L starts iteration in round $p+1$, $\{{}^{w}\boldsymbol{X}_{j}^{(p+1)}, j \in N(i)\cap lay_{L-1}\}$ and $\{{}^{w}\boldsymbol{X}_{j}^{(p)}, j \in N(i)\cap lay_L \cap lay_{L+1}\}$ are available at it. From the global view, neighbors in the even (odd) layers listen to the updated states of the odd (even) layer neighbors, and all nodes in even (odd) layers update their states in parallel. Fig. 7a shows one example of RIPPLE partly and Fig. 7b explains the iterative process from the time dimension. This process repeats until states converge.

The cost function of RIPPLE scheme is therefore can be written as:

$$R(i) = \sum_{j,k\in N(i),j,k\in lay_{L-1}} R(j^{(p+1)},k^{(p+1)}) + \sum_{j,k\in N(i),j\in lay_{L-1},k\in lay_L\cup lay_{L+1}} R(j^{(p+1)},k^{(p)}) + \sum_{j,k\in N(i),j,k\in lay_L\cup lay_{L+1}} R(j^{(p)},k^{(p)}) \quad (13)$$

where node i is in the layer L.

When the error function change of a UAV is less than a preset threshold, the UAV can stop iteration and get the final state results.

C. Analysis of the Iteration Schemes

Generally, the format of the message sent by the UAV i after iteration $p+1$ is defined as $[StateRefresh\_cmd, ID = i, {}^{w}\boldsymbol{X}_{i}^{(p+1)}, p+1]$. The message itself has a constant size. We mark its size as $O(M)$. For the RIPPLE iteration scheme and the other two, the condition for each UAV to stop iteration is that the difference of the objective function is less than a certain threshold. From then on, the UAV stops executing the local graph optimization and sending StateRefresh_cmd. Suppose the UAV group stops the algorithm after P iterations. Therefore, the communication cost per UAV is $O(PM)$.

As the iteration order in the three methods is different, the total time spent by the system and the communication cost to execute the algorithm are also different. Assume that the computation time for a UAV to execute one round of local graph optimization algorithm is $O(d)$. The time to transmit the state data is $O(s)$. Suppose the maximum node degree is Δ. The maximum number of neighbors in the same layer is denoted by Δ_L. It is obvious that $\Delta_L < \Delta$.

In the sequential method, all nodes must be executed sequentially in order, so the time is $O(PN(d+s))$. In contrast, the time cost of the parallel method is $O(P(d+\Delta s))$. Δs is the updated state transmission time considering time-division collision avoidance with neighbors. For the RIPPLE scheme, suppose the number of layers of the network is h. Then the time of RIPPLE scheme is $O\left(2(P-1)(d+\Delta_L s)+h(d+\Delta_L s)\right)$. This can be seen from the explanation in Fig. 7a. After the

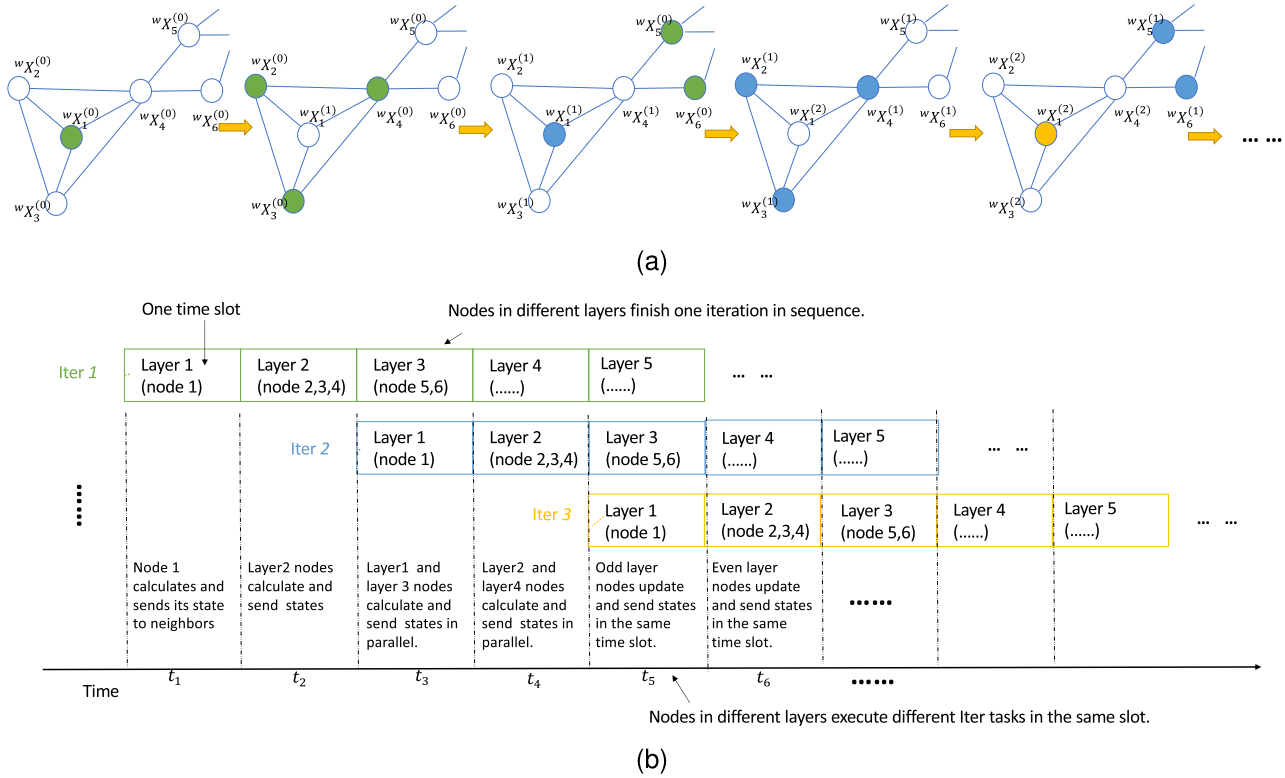

Fig. 7. One example of the RIPPLE iteration strategy. In (a), node 1 is the first layer. Nodes 2, 3, and 4 are in the second layer and nodes 4 and 5 are in the third layer. A colored UAV node indicates that it is executing the algorithm. Different colors represent different rounds of iteration. In (b), the numbers in the rectangles represent which layer is executing the algorithm. For example, at time t_1, layer 1 is executing the first iteration. At t_5, node 1 starts the third iteration; layer 3 nodes are updating their states in the second iteration, while layer 5 nodes are executing their first iteration.

first h iterations, the odd and the even layer nodes update states interchangeably for $2(P-1)$ iterations. So the total time of RIPPLE is proportional to P and h. which is comparable with the parallel method and is much less than the sequential method.

VI. Experiments

A. Experiment Setup

We conduct experiments on a widely used multiple-UAV simulation platform, i.e., AirSim [23]. AirSim provides the ground truth of UAVs' states, so it is very convenient for evaluating the state estimation accuracy. Moreover, large-scale UAV clusters can be simulated on the simulation platform. In the experiment, we have adopted a series of experimental parameters that are in line with reality. The experiment scenarios are simulation environments provided by AirSim, which are available at https://github.com/microsoft/AirSim/releases. Each UAV is simulated to equip with a UWB-based ranging device and a mono-camera mounted on the UAV's front side. The camera takes images at 10Hz, and the image size is 960*540 pixels. Local image processing is conducted for each captured image to extract ORB feature points in the image. At the same time, YOLO-lite [21] based target detection algorithm is carried out to extract the bounding boxes of other UAVs if they appear in the image of the current UAV. The maximum ranging distance of the UWB device is set to 100m, and random Gaussian noises are added to the ranging measurements. The distance measurements are assumed symmetric.

For setting up common view measurement edges, ORB feature brute-force matching is conducted when a UAV receives ORB feature points from its neighbor. A common view edge is set up only when more than 15 pairs of feature points are matched between the two UAVs. The relative poses between the two UAVs are calculated by the matched feature points using 2D-2D geometry, which is used as the common view measurement edge (Equation 6 and 7).

In the experiments, UAVs are time-synchronized. We use LCM (Lightweight Communications and Marshalling[1]) [56] to simulate efficient and low latency communication between UAV nodes. The UAVs are simulated to fly in a 3D environment, take measurements and update relative states round by round. Since we focus on relative state estimation by distributed graph optimization, the UAVs have not used odometry information to update their states. The odometry information can be incorporated in practical applications.

In the experiment evaluations, we use RMSE(Root Mean Square Error) [57] metric to evaluate the translation and rotation error. The translation error of the whole system will be calculated by:

$$ {}^{t}E = \sqrt{\sum_{{}^{w}\boldsymbol{X}_i \in X} (({}^{w}\hat{\boldsymbol{X}}_i)_t - ({}^{w}\boldsymbol{X}_i)_t)^2 / N} \tag{14} $$

where ${}^{t}\hat{\boldsymbol{X}}_i$ is the translation part of the state estimation result and ${}^{t}\boldsymbol{X}_i$ is the translation part of the ground truth.

[1]https://github.com/lcm-proj/lcm

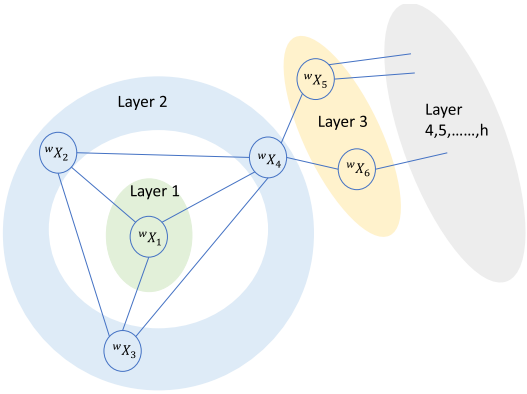

Fig. 8. Nodes are divided into corresponding layers. Node 1 is in the first layer, nodes 2, 3, and 4 are in the second layer, and the third layer contains nodes 4 and 5. Other nodes continue to be divided according to the same rules.

Similarly, the rotation error is defined as:

$$^{R}E = \sqrt{\frac{1}{N}\sum_{\boldsymbol{X_i}\in\boldsymbol{X}}\sum_{s=0,1,2}((^{w}\hat{\boldsymbol{X}}_{\boldsymbol{i}})_{\boldsymbol{R_s}} - (^{w}\boldsymbol{X_i})_{\boldsymbol{R_s}})^2} \quad (15)$$

where R_0, R_1, R_2 represent the rotation angles of the UAV in three orthogonal directions. The translation error is in meters and the rotation error is in degrees.

B. Initialization Performances of Measurement Graphs Using Different Local Graph Models

In the very beginning, only UAV 1 knows its states, which is used as the reference coordinate system. In the LTI initialization method, the UAVs take measurements, share the measurements with neighbors using the $Init\_Cmd$, and initialize their relative poses to UAV 1. For the entire system, initialization needs to be performed only once. Subsequent iterations can take the result of the previous iteration as the initial value.

One of the key functions in the initialization phases is each UAV sends its locally processed measurement data to neighbors, so that each UAV can establish its local graph model based on the newly captured measurements of itself and that of its neighbors. The local graph model is the basis of iterative graph optimization. The differences of the constructed local graphs in NBM and PBM are compared when the number of UAVs is different.

The average number of different types of edges in the constructed local graphs in these two models are compared in Table I. The average number of the edges in PBM is obviously higher than that in NBM. The last column compares the local measurement graph construction time. PBM uses slightly more time than NBM and both of them have good real-time performance. Since PBM contains more more local information, we use PBM model in the following experiments.

C. Distributed Graph Optimization

For testing the performances of different distributed graph optimization methods, we combine the PBM model with three distributed iteration schemes, i.e., RIPPLE, sequential, and parallel. The performance of the centralized method is also considered. We also compare our method with the two SOTA distributed graph optimization methods. The compared methods are briefly introduced as follows:

TABLE I
COMPARISON OF INITIALIZATION WITH DIFFERENT NUMBERS OF UAVS

#UAVs	Model	Average Number of Measurement Edges			Time(ms)
		Distance	Object-detection	Common-view	
10	NBM	3.8	2.5	2.1	10
	PBM	6	4.1	3.5	10
20	NBM	3.95	2.85	2.5	14
	PBM	9	3.9	4	15
100	NBM	4.1	3.0	2.5	34
	PBM	9	3.9	4	39

1) **Centralized:** All data are collected to a central server and centralized graph optimization is applied.
2) **PBM + Parallel:** The local graph contains the edges with the direct neighbors and the edges between the direct neighbors. Parallel iteration scheme is adopted.
3) **PBM + Sequential:** The local graph contains the edges with the direct neighbors and the edges between the direct neighbors. Sequential iteration scheme is adopted.
4) **PBM + RIPPLE:** The local graph contains the edges with the direct neighbors and the edges between the direct neighbors. RIPPLE iteration scheme is adopted.
5) **DGS [43]:** DGS is a state-of-the-art synchronous distributed graph optimization method, which is also used in the multi-robot SLAM system [45]. In DGS, the nonlinear graph optimization problem is transformed into a linear system, solved by the distributed Gauss Seidel method.
6) **ASAPP [17]:** Asynchronous Stochastic Parallel Pose Graph Optimization (ASAPP) is the first asynchronous algorithm for distributed graph optimization in multi-robot localization. ASAPP uses Riemannian optimization to solve the rank-restricted relaxations of graph optimization and applies Poisson clock to control asynchronous iteration.

1) Accuracy Performance: At first, we compare the accuracy performances of different distributed methods. The results of the average translation error, rotation error, and time-consuming of the algorithm are shown in Table II. The data are all from the Airsim simulation platform. The basic settings are the same with those in VI-A. In ASAPP, the stepsize is $\gamma = 5 \times 10^{-4}$ and the rate parameter is $\lambda = 1000$, which are the same as in its paper.

As can be seen from the results, under the circumstance of no global information for each UAV, the estimation results of distributed methods are comparable to the centralized approach. In comparing of the iteration schemes, the efficiency and accuracy of the three strategies are different. Generally speaking, the parallel scheme is the fastest, and the sequential scheme is the slowest. However, the sequential scheme generates a more accurate pose estimation than the parallel scheme. The RIPPLE scheme performs well in both speed and accuracy. Its time latency is comparable to the parallel scheme, which is much shorter than the sequential scheme. Its accuracy is also as good as the sequential method. Among the

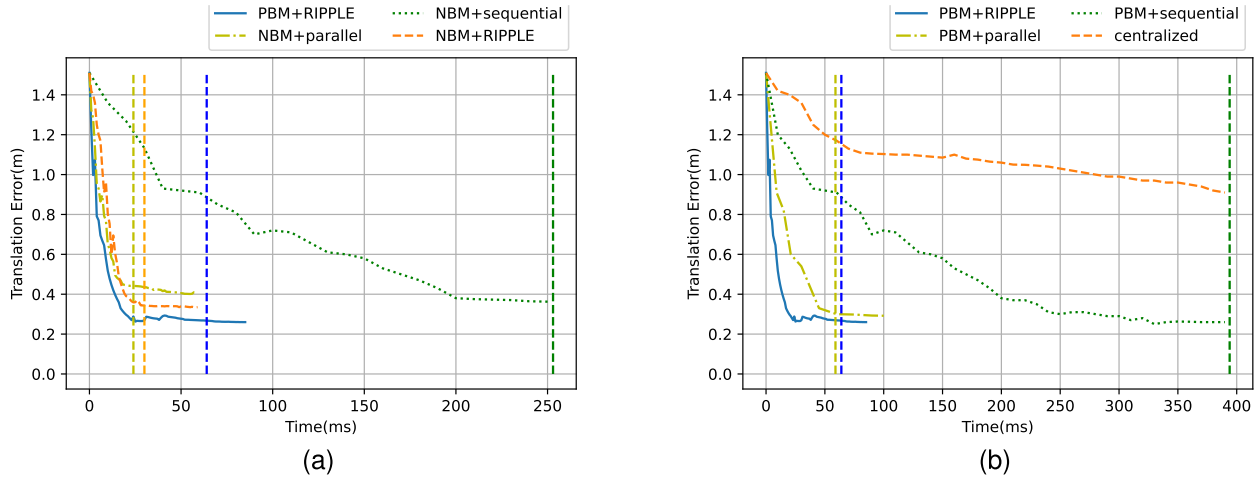

Fig. 9. Convergence speed of different methods on 20 UAVs. For convenience of observation and comparison, the convergence results of PBM + RIPPLE method and other baselines are shown in (a) and (b) respectively. The vertical dotted line indicates that the error change of the corresponding distributed algorithm is small enough to stop the iteration.

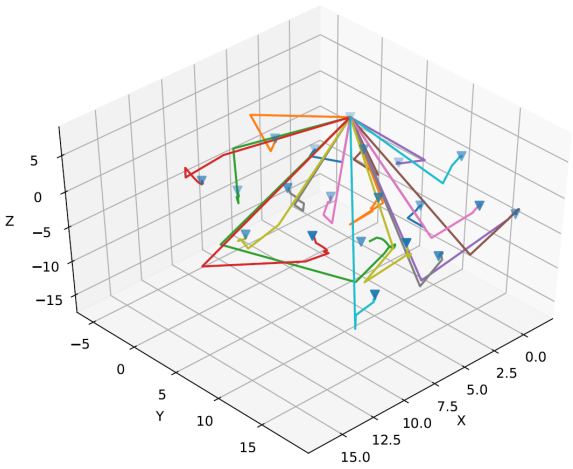

Fig. 10. The convergence trails of PBM + RIPPLE method with 20 UAVs.

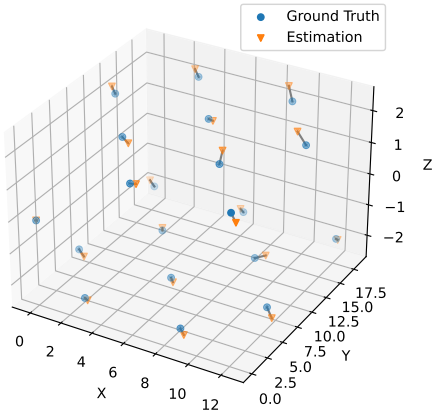

Fig. 11. The convergence result of the PBM + RIPPLE algorithm for 20 UAVs at a certain time. The ground truth and the estimation value are connected by a straight line.

methods proposed in this paper, the PBM + RIPPLE method achieves the best balance between localization accuracy and low latency.

Compared with DGS and ASAPP, in the small-scale cluster of 5 UAVs, DGS and ASAPP perform best. The results of PBM + RIPPLE method are not in the first place but still comparable with the best methods. However, in large-scale clusters, such as when there are more than 10 UAVs, PBM + RIPPLE algorithm performs better than DGS and ASAPP in accuracy and time.

2) Convergence Process: Fig. 10 shows the convergence trails of PBM + RIPPLE with 20 UAVs. The UAVs' relative states change quickly in the first several iterations. They converge fast to be close to the ground truth. The comparison of the ground truth and the convergence result is shown in Fig. 11.

The convergence speed is a critical performance for UAV network relative state estimation. A comparison of the convergence speed and accuracy of different methods is shown in Fig. 9a and Fig. 9b. For 20 UAVs, there is nearly no difference in the convergence speed between the RIPPLE iteration and the parallel iteration schemes. Nevertheless, PBM + RIPPLE provides better accuracy than PBM + Parallel. Sequential iteration and centralized method need more time to converge. Besides, the accuracy of the method using PBM is higher than that using NBM.

3) Communication Analysis: Communication efficiency is also critical as it plays a vital role in low latency and quick convergence. The measurement information has been sent once for each node in the initialization phase in each round. In the iteration process of distributed graph optimization, UAV nodes only send and receive state information among neighbors. The node will receive information, update the state, and then send the latest state to neighbors in each iteration.

Similar to the communication experiment during initialization, we count the number of packets sent by each UAV in the distributed iteration phase, which is treated as the communication cost. The communication costs of three iteration schemes are compared, and the results are shown in Fig. 12.

The communication cost is determined by the number of iterations to a certain extent because the UAV needs to send it in each iteration. As the RIPPLE method can converge more quickly than the parallel method, it uses less communication

TABLE II
COMPARISON OF RELATIVE STATE ESTIMATION OF FIVE KINDS OF DISTRIBUTED METHODS AND CENTRALIZED METHOD WITH DIFFERENT NUMBERS OF UAVS

#Num	Method	${}^{t}E$(m)	${}^{R}E$(degree)	Time(ms)
5	Centralized	0.1	5.5	30
	DGS	0.08	**1.8**	14
	ASAPP	**0.07**	2.0	**12**
	PBM Parall	0.24	5.4	16
	PBM Sequ	0.11	4.9	29
	PBM RIPPLE	0.16	5.0	17
10	Centralized	**0.16**	8.1	208
	DGS	0.24	6.4	**20**
	ASAPP	0.22	**5.4**	26
	PBM Parall	0.29	7.6	23
	PBM Sequ	**0.21**	6.2	71
	PBM RIPPLE	**0.21**	6.2	**20**
20	Centralized	**0.18**	7.5	2167
	DGS	0.55	**6.9**	364
	ASAPP	0.35	**6.9**	89
	PBM Parall	0.3	7.1	**59**
	PBM Sequ	**0.26**	7.0	389
	PBM RIPPLE	**0.26**	**6.9**	64
100	Centralized	**0.26**	**7.4**	13404
	DGS	0.61	9.1	2305
	ASAPP	0.59	8.8	247
	PBM Parall	0.39	7.5	**206**
	PBM Sequ	**0.37**	7.5	19614
	PBM RIPPLE	**0.37**	**7.4**	219

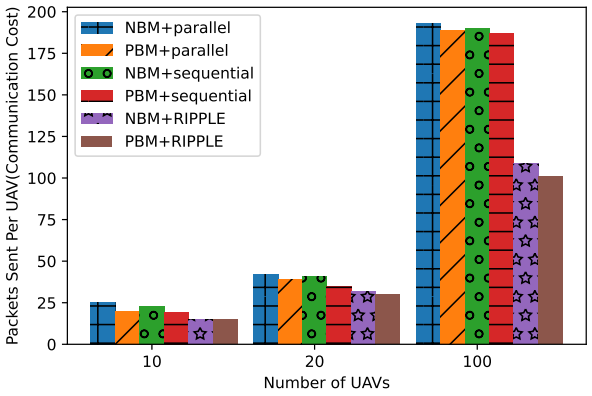

Fig. 12. Comparison of Communication Costs of Different Methods.

TABLE III
COMPARISON OF RELATIVE STATE ESTIMATION WITH 10 UAVS WITH PATCH-BASED MODEL AND RIPPLE ITERATION SCHEME

Method	Trans.Error(m)	Rot.Error(degree)
Proposed Method	0.21	6.2
Without UWB	0.38	7.8
Without Detection	0.28	10.6
Without Common View	0.37	11.1

than the parallel method. In RIPPLE, the UAV does not need to wait for neighbors with smaller ID, so it is also more efficient in communication time than the sequential method.

D. Localization Robustness Performances

We also evaluate the robustness of the proposed methods. We simulate the cases when some kind of sensor data cannot be collected. 10 UAVs are used in the experiments, and PBM + RIPPLE schemes are chosen. The topology of the network is the same as that in the experiment on accuracy performance(VI-C.1). Table III shows the average performances of relative state estimation when one type of particular measurement edge is absent. It shows the robustness of the distributed graph optimization method regarding different combinations of the sensing data. In the experiments, when one kind of observation is missed, the distributed graph optimization model can still converge although the accuracy is reduced. In case of losing the UWB measurements, it has less impact on the rotation error but has an evident impact on the transition error. Losing the target detection measurements impact more on the rotation error. The absence of common-view edges impacts both the transition and rotation errors. These results are consistent with our common sense.

E. Localization Performance vs. Network Sparsity

Another situation that needs to be analyzed is how the distributed optimization methods perform when the the network is sparse. In order to evaluate different methods' performances in different degrees of network sparsity, we conduct an experiment using 20 UAVs with the RIPPLE scheme. Starting from a dense network, each UAV just takes random Brownian motions. This is simulated by letting each UAV take a Gaussian distributed random motion (zero mean, t variance) in each step. As time passes, the UAVs will become scattered and the network becomes sparser and sparser.

Fig. 13a, Fig. 13b, and Fig. 13c show the network topologies in three different time slots, in which the UAV network becomes more and more sparse. When the distances among UAVs increase, the local measurement graph of each UAV

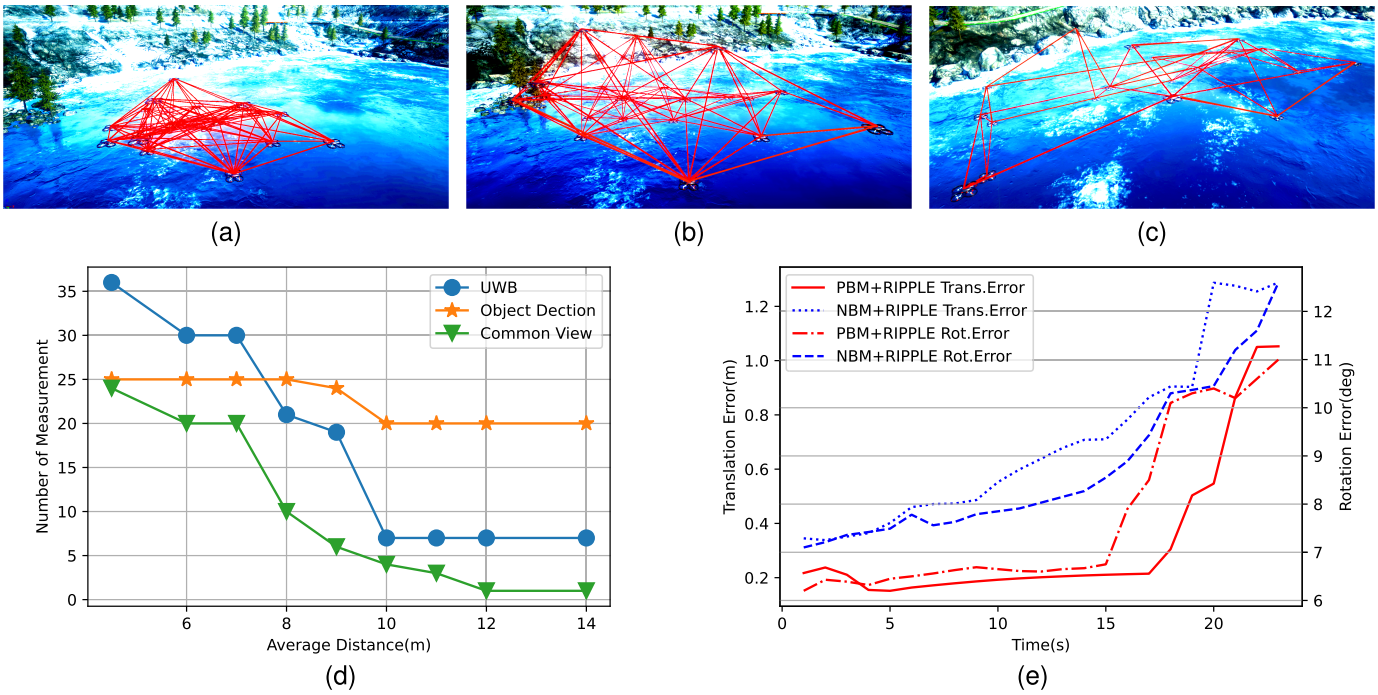

Fig. 13. (a), (b) and (c) are the UAV networks in the simulation scene. The red line indicates measurement and communication between the two UAVs. The UAVs in (a)(b)(c) move more and more away from each other, making the distance between UAVs expand continuously. (d) is the number of measurements as the average distance changes; (e) shows the translation and rotation errors when the network becomes sparse over time.

become sparser for both the NBM and PBM models. The measurements from both UWB and environment common view will decrease significantly, which is shown in Fig. 13d. The object detection measurements change slightly which comes from the fact that some UAVs may leave the field of view when the distance increases.

As time passes, we evaluate the relative state estimation accuracy of different methods to evaluate their performances when the network becomes sparser. The results are shown in Fig. 13e. NBM + RIPPLE and PBM + RIPPLE methods are compared.

From the results, both the rotation accuracy and transition accuracy of NBM + RIPPLE method increase with the network sparsity, as shown by the blue dashed lines. For PBM + RIPPLE, the rotation error and the translation error increase only slightly when t is small, until t is around 15s, at which the number of common view measurements and the number of UWB measurements decrease sharply. This results at a significant increase of the rotation and translation errors. Overall, PBM + RIPPLE method shows better tolerance against network sparsity than NBM + RIPPLE method.

F. Localization Performance vs. GNSS Initial Values

The previous discussions and experiments are based on the premise that there is no global positioning information. However, in practical application, a few UAVs may be equipped with GNSS(Global Navigation Satellite System). This part of UAVs will have better initial values. Different numbers of GNSS initialized UAVs will affect the convergence results. We simulate to provide better initial values (GNSS values) for different numbers of UAVs in the experiment (Fig. 14). The GNSS values are simulated by adding Gaussian noises (zero mean, 0.1 variance) to the ground-truth location. The PBM + RIPPLE method was tested in a network of 20 UAVs in the experiment, and the experiment's setting was the same as that in VI-A. The experiment results show that with the increase in the number of GNSS initialized nodes, the positioning accuracy and the convergence speed can be both improved. However, the accuracy is no longer improved when the number reaches a certain threshold, which is mainly due to the initialization and online measurement noises.

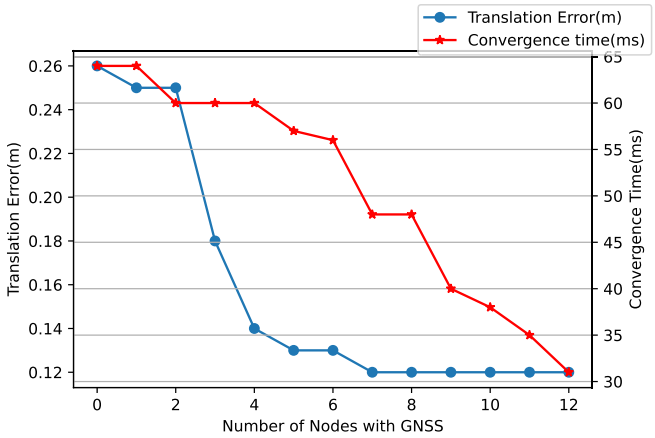

Fig. 14. Comparison of Different Numbers of UAVs with GNSS.

G. Real World Experiments

1) Experiment Settings: To test the algorithm more comprehensively, we carried out a real-world experiment. A network composed of eight nodes is tested. Each node consists of a Jetson nano, a UWB module, and an Astra Pro camera.

TABLE IV
COMPARISON OF RELATIVE STATE ESTIMATION IN THE REAL WORLD EXPERIMENT

Method	Trans.Error(m)	Rot.Error(deg)	Time(ms)
Centralized	0.15	6	140
DGS	0.35	9.1	101
ASAPP	0.37	9.8	105
PBM RIPPLE	0.35	8.4	96

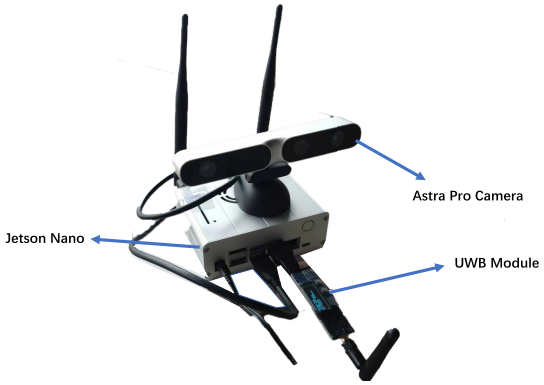

Fig. 15. The node containing a Jetson nano, UWB module, and Astra Pro camera.

Jetson nano is a low-power computer used for computing and communication, which has a 4-core CPU. The UWB module is used to measure the distance to other nodes, and the camera is used to collect RGB images (without depth). We use ROS(Robot Operating System) to carry out the internode communication, and the time synchronization is realized in ROS by Timesynchronizer,[2] so that each node can uniformly receive messages delivered by various topics.

Limited by equipment, we conducted the experiment when nodes were static to validate the convergence speed and accuracy of the different algorithms. The ground truth values of each node are calibrated by laser ranging and manual measurement. The eight nodes form one cluster. They collected the UWB ranging and image data by subscribing to ROS topics of other nodes. This constructs a dataset collected from static nodes in the real environment. The experiment environment is shown in Fig. 16 and the Nano node is shown in Fig. 15. The process of setting up the common-view edge between two Nano nodes using their captured images is shown in Fig. 17.

2) Experiment Results: We then process the dataset to let each node conduct distributed graph optimization and state updating as if they are collecting the data in real time. The localization accuracy after convergence is compared with the calibrated ground truth. The results are shown in Table IV.

In actual experiments, we pay special attention to the accuracy and stability of the sensors. Since accuracy of UWB ranging is higher than the reliability of vision in real experiments, the weight value of the error $R_{uwb}(i,j)$ is increased to improve the localization accuracy. It can be seen that the results obtained from the real-world dataset are analogous to the simulation results since the simulations in AirSim are also based on real-time image processing. PBM + RIPPLE shows slightly better localization accuracy than other methods, which mainly benefited from PBM for it uses more edge constraints. Since the network is small, RIPPLE contributes little to the convergence speed.

[2]http://docs.ros.org/en/api/message_filters/html/c++/classmessage__filters_1_1TimeSynchronizer.html

Fig. 16. The experiment environment in real world.

Fig. 17. The example of common features and object detection in the real-world experiment.

VII. CONCLUSION

This paper investigates the communication efficient, distributed relative state estimation method in UAV networks. We combine distance measurement, object detection, and feature points matching together with only UWB and cameras onboard to obtain the relative states of UAVs using distributed graph optimization. A local graph model, i.e., patch-based model (PBM), is proposed. A communication-efficient initialization method (LTI, Locating Then Informing) is proposed to help each UAV obtain the initial state in a highly efficient way and construct the local graph. We have also proposed a novel distributed iteration scheme, i.e., RIPPLE, for communication efficient distributed graph optimization. Experiments by AirSim and experiments on Nano devices, show that the distributed graph optimization method provides accurate results and is comparable to the centralized method. PBM + RIPPLE outperforms other combinations of local graph models and iteration schemes in terms of accuracy, low latency, and robustness. In the future, we will further develop the experiment system to verify the relative state estimation performances when the nodes are dynamic. Then we will consider using the information from more sensors, such as the Inertial Measurement Unit (IMU), to further enhance the distributed relative state estimation performances.

ACKNOWLEDGMENT

The authors would like to thank the reviewers' valuable suggestions to improve this work and also would like to thank Prof. Qianchuan Zhao of Tsinghua University for providing space and guidance in conducting the real-world experiments.

REFERENCES

[1] L. A. Villas, D. L. Guidoni, and J. Ueyama, "3D localization in wireless sensor networks using unmanned aerial vehicle," in *Proc. IEEE 12th Int. Symp. Netw. Comput. Appl.*, Aug. 2013, pp. 135–142.

[2] S. Goel et al., "Cooperative localization of unmanned aerial vehicles using GNSS, MEMS inertial, and UWB sensors," *J. Surveying Eng.*, vol. 143, no. 4, Nov. 2017, Art. no. 04017007.

[3] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein, "Real-time vision-aided localization and navigation based on three-view geometry," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 3, pp. 2239–2259, Jul. 2012.

[4] C. Shinde, R. Lima, and K. Das, "Multi-view geometry and deep learning based drone detection and localization," in *Proc. 5th Indian Control Conf. (ICC)*, Jan. 2019, pp. 289–294.

[5] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli, "GOMSF: Graph-optimization based multi-sensor fusion for robust UAV pose estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1421–1428.

[6] B. Dai, Y. He, L. Yang, Y. Su, Y. Yue, and W. Xu, "SIMSF: A scale insensitive multi-sensor fusion framework for unmanned aerial vehicles based on graph optimization," *IEEE Access*, vol. 8, pp. 118273–118284, 2020.

[7] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized visual-inertial-UWB fusion for relative state estimation of aerial swarm," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 8776–8782.

[8] B. Anderson, B. Fidan, C. Yu, and D. Walle, "UAV formation control: Theory and application," in *Recent Advances in Learning and Control*. London, U.K.: Springer, 2008, pp. 15–33.

[9] D. R. Green, J. J. Hagon, C. Gómez, and B. J. Gregory, "Using low-cost UAVs for environmental monitoring, mapping, and modelling: Examples from the coastal zone," in *Coastal Management*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 465–501.

[10] H. Xu et al., "Omni-swarm: A decentralized omnidirectional visual-inertial-UWB state estimation system for aerial swarm," 2021, *arXiv:2103.04131*.

[11] R. Chen, B. Yang, and W. Zhang, "Distributed and collaborative localization for swarming UAVs," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 5062–5074, Mar. 2020.

[12] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5220–5227.

[13] T. Han, Z. Lin, R. Zheng, Z. Han, and H. Zhang, "A barycentric coordinate based approach to three-dimensional distributed localization for wireless sensor networks," in *Proc. 13th IEEE Int. Conf. Control Autom. (ICCA)*, Jul. 2017, pp. 600–605.

[14] Y. Liu, Y. Wang, J. Wang, and Y. Shen, "Distributed 3D relative localization of UAVs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 11756–11770, Oct. 2020.

[15] B. Hwa Cheng, R. E. Hudson, F. Lorenzelli, L. Vandenherghe, and K. Foo, "Distributed Gauss-Newton method for node loclaization in wireless sensor networks," in *Proc. IEEE 6th Workshop Signal Process. Adv. Wireless Commun.*, Jun. 2005, pp. 915–919.

[16] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using constrained factor graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 3025–3030.

[17] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and parallel distributed pose graph optimization," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5819–5826, Oct. 2020.

[18] Q. Yang, X. Yang, Y. Yang, and Y. Yang, "Distributed cooperative localization based on bearing-only sensors," *IEEE Sensors J.*, vol. 21, no. 20, pp. 23645–23657, Oct. 2021.

[19] 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," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3349–3355.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[21] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A real-time object detection algorithm optimized for non-GPU computers," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2503–2510.

[22] A. S. Bedi, A. Koppel, and K. Rajawat, "Asynchronous saddle point method: Interference management through pricing," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 3229–3235.

[23] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. Cham, Switzerland: Springer, 2018, pp. 621–635.

[24] T. M. Nguyen, A. H. Zaini, K. Guo, and L. Xie, "An ultra-wideband-based multi-UAV localization system in GPS-denied environments," in *Proc. Int. Micro Air Vehicles Conf.*, 2016, pp. 1–6.

[25] T. Ziegler, M. Karrer, P. Schmuck, and M. Chli, "Distributed formation estimation via pairwise distance measurements," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3017–3024, Apr. 2021.

[26] 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," *Int. J. Micro Air Vehicles*, vol. 9, no. 3, pp. 169–186, Sep. 2017.

[27] K. Guo, X. Li, and L. Xie, "Ultra-wideband and odometry-based cooperative relative localization with application to multi-UAV formation control," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2590–2603, Jun. 2019.

[28] B. Jiang, M. Deghat, and B. D. O. Anderson, "Simultaneous velocity and position estimation via distance-only measurements with application to multi-agent system control," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 869–875, Feb. 2016.

[29] S. Yang, S. A. Scherer, and A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle," *J. Intell. Robotic Syst.*, vol. 69, nos. 1–4, pp. 499–515, Jan. 2013.

[30] M. Saska et al., "System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization," *Auton. Robots*, vol. 41, no. 4, pp. 919–944, 2017.

[31] N. Piasco, J. Marzat, and M. Sanfourche, "Collaborative localization and formation flying using distributed stereo-vision," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1202–1207.

[32] M. W. Achtelik, S. Weiss, M. Chli, F. Dellaerty, and R. Siegwart, "Collaborative stereo," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 2242–2248.

[33] G. T. F. de Abreu and G. Destino, "Super MDS: Source location from distance and angle information," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2007, pp. 4430–4434.

[34] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proc. 4th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2003, pp. 201–212.

[35] Y. Wang, T. Sun, G. Rao, and D. Li, "Formation tracking in sparse airborne networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2000–2014, Sep. 2018.

[36] H. Ping, Y. Wang, D. Li, and T. Sun, "Flipping free conditions and their application in sparse network localization," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 986–1003, Mar. 2022.

[37] S. S. Kia, S. Rounds, and S. Martinez, "Cooperative localization for mobile agents: A recursive decentralized algorithm based on Kalman-filter decoupling," *IEEE Control Syst. Mag.*, vol. 36, no. 2, pp. 86–101, Apr. 2016.

[38] S. Safavi, U. A. Khan, S. Kar, and J. M. F. Moura, "Distributed localization: A linear theory," *Proc. IEEE*, vol. 106, no. 7, pp. 1204–1223, Jul. 2018.

[39] F. Wu, Y.-P. Tian, and B. Wang, "Distributed localization in sensor networks with noisy distance measurements," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 8917–8922.

[40] H. Ping, Y. Wang, D. Li, and W. Chen, "Understanding node localizability in barycentric linear localization," *IEEE/ACM Trans. Netw.*, early access, Nov. 8, 2022, doi: 10.1109/TNET.2022.3216204.

[41] L. Buşoniu, R. Babuška, and B. D. Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Application-1*. Berlin, Germany: Springer, 2010, pp. 183–221.

[42] B. Peng, G. Seco-Granados, E. Steinmetz, M. Frohle, and H. W. Wymeersch, "Decentralized scheduling for cooperative localization with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4295–4305, May 2019.

[43] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed Gauss–Seidel approach," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 5261–5268.

[44] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *Int. J. Robot. Res.*, vol. 36, no. 12, pp. 1286–1311, Oct. 2017.

[45] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1656–1663, Apr. 2020.

[46] Y. Chang, Y. Tian, J. P. How, and L. Carlone, "Kimera-multi: A system for distributed multi-robot metric-semantic simultaneous localization and mapping," 2020, *arXiv:2011.04087*.

[47] S. Vashisht, S. Jain, and G. S. Aujla, "MAC protocols for unmanned aerial vehicle ecosystems: Review and challenges," *Comput. Commun.*, vol. 160, pp. 443–463, Jul. 2020.

[48] A. Jiang, Z. Mi, C. Dong, and H. Wang, "CF-MAC: A collision-free MAC protocol for UAVs ad-hoc networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[49] D. Djenouri and M. Bagaa, "Synchronization protocols and implementation issues in wireless sensor networks: A review," *IEEE Syst. J.*, vol. 10, no. 2, pp. 617–627, Jun. 2014.

[50] K. Skiadopoulos et al., "Synchronization of data measurements in wireless sensor networks for IoT applications," *Ad Hoc Netw.*, vol. 89, pp. 47–57, Jun. 2019.

[51] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 11, pp. 3069–3087, 2021.

[52] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, and J. Barreto, "Camera models and fundamental concepts used in geometric computer vision," *Found. Trends Comput. Graph. Vis.*, vol. 6, pp. 1–183, Jan. 2011.

[53] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[54] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613.

[55] T. H. Nguyen, T.-M. Nguyen, and L. Xie, "Flexible and resource-efficient multi-robot collaborative visual-inertial-range localization," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 928–935, Apr. 2021.

[56] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 4057–4062.

[57] G. Casella and R. L. Berger, *Statistical Inference*. Boston, MA, USA: Cengage Learning, 2021.

Shuo Wang received the B.S. degree from the School of Information, Renmin University of China, where he is currently pursuing the Ph.D. degree. His research interests include network localization algorithms and graph optimization and applications.

Yongcai Wang (Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Automation Sciences and Engineering, Tsinghua University, in 2001 and 2006, respectively. He worked as an Associate Researcher at the NEC Laboratories, China, from 2007 to 2009. He was a Research Scientist with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, from 2009 to 2015. He was a Visiting Scholar at Cornell University in 2015. He is currently an Associate Professor at the School of Information, Renmin University of China. His research interests include algorithms for large graph mining and structure realization, algorithms for location and mapping in robot, and UAV networks.

Xuewei Bai (Graduate Student Member, IEEE) received the B.S. degree from the Department of Computer Science and Technology, North China Electric Power University, in 2020. She is currently pursuing the Ph.D. degree with the Department of Computer Sciences, Renmin University of China. Her research interests include network localization algorithms and graph optimization and applications.

Deying Li received the M.S. degree in mathematics from Huazhong Normal University in 1988 and the Ph.D. degree in computer science from the City University of Hong Kong in 2004. She is currently a Professor with the Department of Computer Science, Renmin University of China. Her research interests include wireless networks, mobile computing, and algorithm design and analysis.