

# VIRAL SLAM: Tightly Coupled Camera-IMU-UWB-Lidar SLAM

Thien-Minh Nguyen, *Member, IEEE*, Shanghai Yuan, Muqing Cao,  
Thien Hoang Nguyen, *Student Member, IEEE* and Lihua Xie, *Fellow, IEEE*

**Abstract**—In this paper, we propose a tightly-coupled, multi-modal simultaneous localization and mapping (SLAM) framework, integrating an extensive set of sensors: IMU, cameras, multiple lidars, and Ultra-wideband (UWB) range measurements, hence referred to as VIRAL (visual-inertial-ranging-lidar) SLAM. To achieve such a comprehensive sensor fusion system, one has to tackle several challenges such as data synchronization, multi-threading programming, bundle adjustment (BA), and conflicting coordinate frames between UWB and the onboard sensors, so as to ensure real-time localization and smooth updates in the state estimates.

To this end, we propose a two stage approach. In the first stage, lidar, camera, and IMU data on a local sliding window are processed in a core odometry thread. From this local graph, new key frames are evaluated for admission to a global map. Visual feature-based loop closure is also performed to supplement loop edges to the global factor graph. When the global factor graph satisfies a condition on spatial diversity, the BA process will be started, and the coordinate transform between UWB and onboard SLAM systems can be estimated. The system then seamlessly transitions to the second stage where all sensors are tightly integrated in the odometry thread. The capability of our system is demonstrated via several experiments on high-fidelity graphical-physical simulation and public datasets.

## I. INTRODUCTION

Localization is arguably one the most important capabilities for mobile robots, especially for Unmanned Aerial Vehicles (UAVs). Obviously, a common approach to ensure reliable and accurate localization is to combine multiple sensors for their complementary advantages as well as redundancy. For example, since lidar is not affected by lighting condition or lack of visual features, which can easily destabilize most visual-inertial-odometry (VIO) systems, the robot can still rely on this type of sensor for localization in low light or low-texture conditions. In addition, camera can also enable loop closure capability, while the lidar pointcloud map can help augment the depth estimation process of visual features [1], [2]. This is one of the main motivations for us to investigate a tightly coupled camera-lidar-IMU-based Simultaneous Localization and Mapping (SLAM) system in this paper.

Besides the aforementioned benefits of a visual-lidar localization system, integration of Ultra-wideband (UWB) to the SLAM system can also provide another layer of backup in case both lidar and camera lose track, and also

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, under the Wallenberg-NTU Presidential Postdoctoral Fellowship Program. (Corresponding author: Thien-Minh Nguyen)

The authors are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, 50 Nanyang Avenue. (e-mail: thienminh.nguyen@ntu.edu.sg, elhxie@ntu.edu.sg)

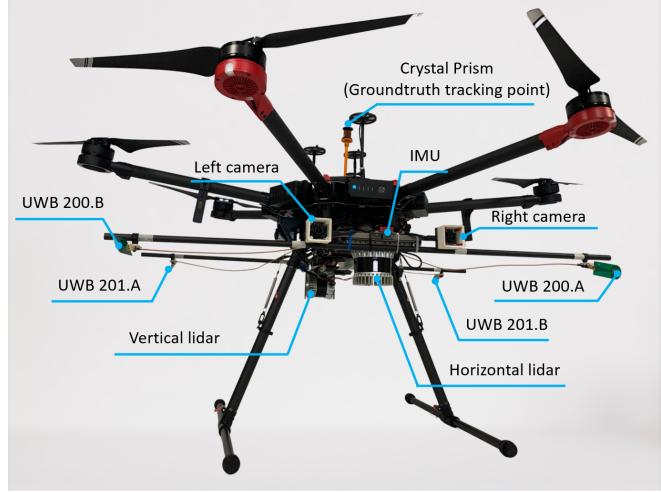


Fig. 1: Hardware setup of the VIRAL SLAM system: a hardware-synchronized stereo camera rig, a 400 Hz IMU, two 16-channel lidars, four body-offset UWB ranging nodes and a crystal prism that is tracked by a Leica total station for millimeter-accuracy groundtruth.

allows user to obtain global localization information relative to the inspected object [3], [4]. However, to successfully integrating UWB with SLAM, especially in the real-time localization process, one must first estimate the coordinates of the anchors in the SLAM coordinate frame  $L$ , which is the methodology used in previous works [5], [6]. In this paper, we propose a new approach. Specifically, using the distance measurements between the anchors, we can set the coordinates of the anchors, which effectively defines a preferred frame  $W$  that aligns with the mission to be conducted in the environment (more details in Sec. III-B). Hence the transform between  $L$  and  $W$  can be refined in the BA process. Subsequently, the anchors' coordinates in  $W$  can be converted to  $L$  and used for constructing the range-based factors in the optimization process over the local sliding window. The separation of estimating the anchor coordinates and estimating the robot states is a deliberate choice to ensure convergence, especially in the case when the movement in the sliding window is too short, which lacks excitation for convergence of the anchor position estimates. It is also noted that we focus on a simple yet effective UWB network of two or three anchors, with multiple body-offset ranging nodes in the UAV. This simple network allows relatively accurate initialization of the robot and anchor position in  $W$ , which facilitates accurate and seamless integration of UWB into

the SLAM system.

The contribution of our work can be stated as follows:

- We propose a comprehensive SLAM framework that tightly integrates multiple sensors of different sensing modalities, i.e. lidars, cameras, IMU and UWB ranging sensors, in a seamless manner.
- We devise a loop closure scheme that is triggered by visual place recognition and further refined via a two-stage pointcloud alignment.
- We propose a novel scheme to fuse UWB, where estimation of the anchor position and ranging bias is delegated to the bundle adjustment (BA) thread, and their values are fixed in the local sliding window optimization for the fusion of UWB range.

## II. RELATED WORKS

To the best of our knowledge, our work features a tightly coupled SLAM system that integrates one of the most comprehensive sensor suites. While localization methods based on mainly camera or lidar (with or without IMU) are abundant, only a handful of works have investigated tightly coupled visual and lidar information in the literature. In [1], Zhang et al proposed a method where VIO and lidar data were employed in a cascaded manner. In this framework, high rate VIO data is used to help initialize the scan-matching process, while pointcloud map can be used to help retrieve the visual feature's depth. On the other hand, in [7], [8] Zuo et al proposed an MSCKF framework to asynchronously update the robot states when lidar and visual features are obtained, and IMU is used for propagation in between of these states. We note that loop closure and BA are not considered in the aforementioned works, therefore drift is still an intrinsic problem in these approaches.

To address the drift issue, in [9], Graeter et al studied the problem of estimating the scale of a pose graph obtained from monocular visual SLAM with lidar information. In [10], Shao et al considered an approach similar to [1], but also used camera and ICP for loop closure. However they require hardware-synchronized camera-lidar messages, which can only produce very low rate data. In [2], Shan et al proposed a loose integration of VIO output from VINS-Mono [11] to LIO-SAM [12], which itself loosely integrates LeGO-LOAM output [13] with IMU preintegration, in a gtsam pose-graph optimization framework. This approach can be unreliable as the whole chain depends on whether the core lidar-based process runs well. If there is a low-texture case when lidar localization is unstable, its error can reverberate up the chain, which appears to be the case in some of our experiments that will be presented in later parts. We also note that the aforementioned works [1], [2], [9], [10], [12], [13] do not consider the integration of multiple lidars. While the MLOAM method [14] addresses this issue, it focuses purely on lidar and no camera and IMU is involved. In [15], we proposed a multi-input lidar-inertia odometry and mapping scheme called MILIOM, which clearly demonstrates the robustness, accuracy, and real-time performance. Our VIRAL SLAM system is based on this lidar-based system.

Another trend in the literature is the use of UWB to aid VIO or SLAM process. For e.g. it has been integrated with monocular VIO for drift correction [5], [6], or can be used as a variable baseline for cameras on different UAVs [16]. In recent years, VIO, UWB and lidar have also been used in a loosely coupled manner for relative localization [17]–[22]. In our previous work [3], tightly coupling of UWB with lidar and IMU preintegration factors were investigated. Though this preliminary work is still restrictive in that the lidar processing pipeline inherited from LIO-Mapping [23] was quite inefficient, and multiple lidar as well as camera were not considered.

Inspired by the aforementioned works above, in this paper we present a comprehensive SLAM method leveraging visual-inertial-ranging-lidar data. For IMU, we employ the preintegration technique [24] to provide observation that couples two consecutive states in the sliding window. We adopt the visual processing part from VINS-Fusion [11], and the loop closure detection part based on DBoW2 [25]. Our lidar feature extraction, feature to map matching, and key frame management schemes are inspired by LIO-SAM [12]. However we include significant modifications to integrate multiple lidars and employ parallel computing to cut down the processing time. The visually-detected lidar-enhanced loop closure process is similar in spirit to [10], though we develop it independently with substantial addition in the relative constraint estimation. We also do not require strict hardware synchronization, which allows estimate to be published at both lidar and IMU rates.

The remainder of the paper is organized as follows: in Sec. III, we lay out the basic definitions, notations, problem formulations, and general approach towards synchronization; Sec. IV then presents the main function blocks that process the sensor data for the local sliding window optimization, while Sec. V goes into detail of the global map management, which includes loop closure and BA processes. We demonstrate the capability of our method via several experiments on public and simulated datasets in Sec. VI. Finally, Sec. VII concludes our work.

## III. PRELIMINARIES

### A. Nomenclature

We use  $(\cdot)^\top$  to denote the transpose of an algebraic vector or matrix under  $(\cdot)$ . For a vector  $\mathbf{x} \in \mathbb{R}^m$ ,  $\|\mathbf{x}\|$  stands for its Euclidean norm, and  $\|\mathbf{x}\|_G^2$  is short-hand for  $\|\mathbf{x}\|_G^2 = \mathbf{x}^\top G \mathbf{x}$ . For two vectors  $\mathbf{v}_1, \mathbf{v}_2$ ,  $\mathbf{v}_1 \times \mathbf{v}_2$  denotes their cross product. In later parts, we denote  $\mathbf{R} \in \text{SO}(3)$  as the rotation matrix, and  $\mathbf{T} \in \text{SE}(3)$  as the transformation matrix. We denote  $\mathbb{Q}$  as the set of unit quaternions, with  $\mathbf{q}$  as its identity element. Given  $\mathbf{q} \in \mathbb{Q}$ ,  $\mathcal{R}(\mathbf{q})$  denotes its corresponding rotation matrix, and  $\text{vec}(\mathbf{q})$  returns its vector part.  $\circ$  denotes the quaternion product,  $\int_{\mathbb{Q}}$  denotes the integration of quaternion derivative. Also, we define the mapping  $\mathcal{E}: \mathbb{R}^3 \mapsto \mathbb{Q}$  and its inverse  $\mathcal{E}^{-1}: \mathbb{Q} \rightarrow \mathbb{R}^3$  to convert a *rotation vector*, i.e. the angle-axis representation of rotation, to its quaternion form, and vice versa. The formulae for these operations can be found in [4].

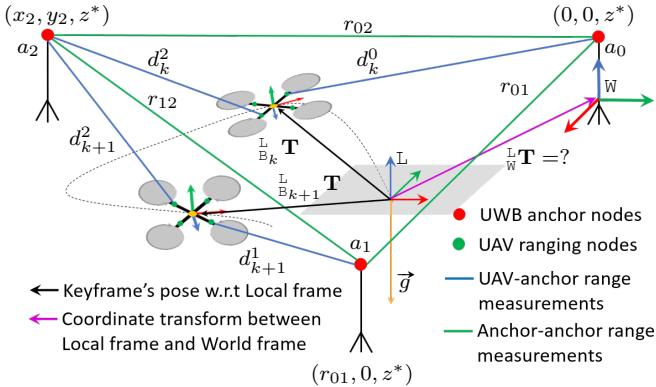


Fig. 2: A so-called world frame  $\mathbb{W}$  can be defined when fixing the coordinates of the anchor nodes using the anchor-to-anchor distances. On the other hand, the SLAM system references to a local coordinate frame  $\mathbb{L}$  that coincides with the initial key frame's pose. To successfully combine UWB with SLAM, the coordinate transform between  $\mathbb{W}$  and  $\mathbb{L}$  must be resolved.

When needed for clarity, we attach a left superscript to an object to denote its coordinate frame. For example  ${}^{\mathbb{A}}\mathbf{v}$  implies the coordinate of the vector  $\mathbf{v}$  is in reference to the frame  $\mathbb{A}$ , and  ${}^{\mathbb{B}}\mathcal{F}$  indicates that the coordinate of the points in the pointcloud  $\mathcal{F}$  is w.r.t. to the frame  $\mathbb{B}$ . A rotation matrix and transformation matrix between two coordinate frames are denoted with the frames attached as the left-hand-side superscript and subscript, e.g.,  ${}^{\mathbb{A}}\mathbb{R}$  and  ${}^{\mathbb{A}}\mathbb{T}$  are called the rotation and transform matrices from frame  $\mathbb{A}$  to  $\mathbb{B}$ , respectively. When the coordinate frames are the body frame at different times, depending on the context, we may omit and rearrange the superscript and subscripts to keep the notation concise, e.g.,  ${}^k\mathbb{R}_{k+1} \triangleq {}^{B_k}_{B_{k+1}}\mathbb{R}$ , or  ${}^w_m\mathbb{T} \triangleq {}^{B_w}_{B_m}\mathbb{T}$ .

Given the position  $\mathbf{p}$ , and the orientation  $\mathbf{q}$  or rotation matrix  $\mathbf{R}$ , we can convert them to transformation matrix and vice versa. Thus with we also write  $\mathbf{T} = (\mathbf{R}, \mathbf{p})$  or  $\mathbf{T} = (\mathbf{q}, \mathbf{p})$ , and refer to  $\mathbf{T}$ ,  $(\mathbf{R}, \mathbf{p})$  or  $(\mathbf{q}, \mathbf{p})$  as the *6DoF pose*, or simply *pose*. Finally, thorough out the paper, when a physical state or a priori quantity has been defined and denoted with non-accented symbol, we may refer to its estimate or measurement by a similar symbol with the hat or breve accents, without formally defining these new quantities, and vice versa. For e.g., if  $\mathbf{q}$  denotes the true orientation state, then  $\hat{\mathbf{q}}$  denotes its orientation estimate, while  $\check{\mathbf{q}}$  denotes its measurement, which could be obtained from IMU propagation.

### B. Coordinate frames

Throughout this paper we define a so-called *local frame*  $\mathbb{L}$  whose origin coincides with the position of the body frame at the initial time (Fig. 2), while  $z$  axis points to the opposite direction of gravity, the  $x$  axis points towards the direction that yields 0-degree yaw for the initial pose of the robot, and  $y$  axis can be inferred from right hand rule. In fact, this coordinate initialization operation is adopted from VINS-Fusion, which uses the initial accelerometer reading

to initialize the roll and pitch angles of the robot, assuming the bias is negligible.

In addition to  $\mathbb{L}$ , we fix the coordinates of the anchors, which defines another so-called *world frame*  $\mathbb{W}$ . Since three anchors reside on a plane, the 3D coordinates of these anchors in  $\mathbb{W}$  can be determined via the distances between the anchors, plus a nominal height  $z^*$ . More specifically, we denote the coordinates of anchors  $a_0, a_1, a_2$  as  $(x_0, y_0, z^*)$ ,  $(x_1, y_1, z^*)$ ,  $(x_2, y_2, z^*)$ . Given the distances between the anchors, denoted as  $r_{01}, r_{02}, r_{12}$  in Fig. 2, using simple geometry we can set  $x_0 = 0, y_0 = 0, x_1 = r^{01}, y_1 = 0, x_2 = \frac{r_{01}^2 - r_{12}^2 + r_{02}^2}{2r_{01}}$  and  $y_2 = \pm\sqrt{r_{02}^2 - x_2^2}$ , where the sign of  $y_2$  determines the direction of the  $y$  axis of  $\mathbb{W}$ .

### C. State estimates

1) *Sliding window*: At each time step  $t_k$ , the robot state  $\mathcal{X}_k$  to be estimated is defined as

$$\mathcal{X}_k = (\mathbf{q}_k, \mathbf{p}_k, \mathbf{v}_k, \mathbf{b}_k^\omega, \mathbf{b}_k^a), \quad (1)$$

where  $\mathbf{q}_k \in \mathbb{Q}$ ,  $\mathbf{p}_k \in \mathbb{R}^3$ ,  $\mathbf{v}_k \in \mathbb{R}^3$  are respectively the orientation quaternion, position and velocity of the robot w.r.t. the local frame  $\mathbb{L}$  at time  $t_k$ ;  $\mathbf{b}_k^a, \mathbf{b}_k^\omega \in \mathbb{R}^3$  are respectively the IMU accelerometer and gyroscope biases, both are referenced in the body frame.

Hence, over a sliding window consisting of the last  $M$  states, we denote the state estimate at each time step, and the sliding window of size  $M$  as follows:

$$\hat{\mathcal{X}}_k = (\hat{\mathbf{q}}_k, \hat{\mathbf{p}}_k, \hat{\mathbf{v}}_k, \hat{\mathbf{b}}_k^\omega, \hat{\mathbf{b}}_k^a), \quad (2)$$

$$\hat{\mathcal{X}} = (\hat{\mathcal{X}}_w, \hat{\mathcal{X}}_{w+1}, \dots, \hat{\mathcal{X}}_k), w \triangleq k - M + 1. \quad (3)$$

Moreover, we also seek to estimate the inverse depth of the features that are tracked on the sliding window, whose state estimated are denoted as:

$$\hat{\lambda}^1, \hat{\lambda}^2, \dots, \hat{\lambda}^{N_\mathcal{V}^k}, \quad (4)$$

where  $N_\mathcal{V}^k$  is the total number of features that are being tracked on the sliding window at time  $t_k$ . More explanation can be found in Sec. IV-B

Note that the indices  $w$  and  $k$  are recurring notations when we refer to the first and last time steps on the sliding window. The lidar extrinsics are not considered in our work since they can be accurately calibrated using one of several tools in the literature, or even measured manually.

2) *Global pose graph and UWB parameters*: A global pose graph is developed with marginalized information from the optimization process on the sliding window. For each key frame  $i$  stored in the memory, we define its pose estimate as  ${}^{\mathbb{L}}\hat{\mathbb{T}}_i$ . The pose estimates will be updated in the BA process whenever a certain number of new key frames are admitted, or a loop factor is obtained.

Besides the key frame pose, we also seek to estimate the following UWB-related parameters:

$${}^{\mathbb{L}}\mathbb{T} = ({}^{\mathbb{L}}\mathbb{R}, {}^{\mathbb{L}}\mathbb{p}), {}^{\mathbb{L}}\mathbb{R} \in \text{SO}(3), {}^{\mathbb{L}}\mathbb{p} \in \mathbb{R}^3; \mathbf{b}^r \in \mathbb{R}, \quad (5)$$

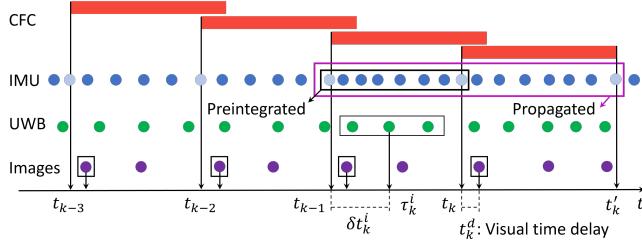


Fig. 3: Synchronization between the CFC and the IMU. The light blue circles represent the interpolated IMU samples.

where  ${}^L_W T$  is the coordinate transform between the local and world frames that were introduced in Sec. III-B, and  $b^r$  is the ranging bias that is present in our problem due to the use of extension cables to place the UWB ranging nodes at different points on the robot [26]. Taking VIO as an analogy, we can see that  ${}^L_W T$  and  $b^r$  are similar to the extrinsic and intrinsic parameters of the UWB ranging and communication network.

#### D. Synchronization

With multiple streams of information from multiple sensors, synchronization of the data is one of the most important issues we need to tackle when building up the SLAM system. The synchronization has to be accurate and efficient to ensure the measurements are coupled with the correct states and all processes adhere to strict real time constraints.

Fig. 3 is an illustration of our synchronization scheme. Among multiple lidars, one is arbitrarily chosen as the primary and others as secondary sources. The timestamp of the primary lidar is used to determine the sliding window's time steps as well as the timestamp of the *combined feature cloud* (CFC) (more details in Sec. IV-A.1). It should be noted that the end time of a CFC can actually be past start time of the next CFC. This affects the synchronization of IMU data, as we have to identify the sequence of samples that are used for preintegration and those used for propagation.

Regarding the UWB samples, the synchronization aims to group them into "bundles" based on the intervals between the time steps. Here we denote the timestamp of a UWB sample as  $\tau_k^i$ , which implies that  $\tau_k^i \in (t_{m-1}, t_m]$ . Knowing this will allow us to associate the sample with the correct state in the construction of the cost factor in the later part.

For the cameras, they are triggered by an external hardware apparatus, thus their images can be synchronized into pairs before further synchronized with the lidar CFCs. For each time step, we will admit the image pair that is closest in time to it, and measure the time delay  $t_k^d$ . This time delay will be used to compensate for the visual feature's pixel coordinate when they are tracked in the image plane.

Fig. 3 is a snapshot of the system at time  $t$ , where all of the sensor data needed for constructing the cost function in the local sliding window optimization block are available. After the optimization process elapses, we can obtain optimized states  $\hat{\mathcal{X}}_w, \hat{\mathcal{X}}_{w+1}, \dots, \hat{\mathcal{X}}_k$  and nominate one of these as a key frame candidate  $\mathcal{K}$  to the global map. This is the snapshot

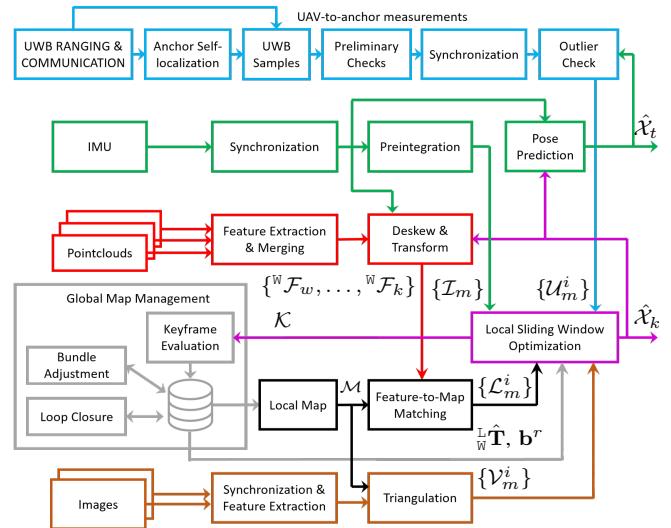


Fig. 4: Overview of the VIRAL SLAM system. The colors are used to distinguish lines that intersect each other but do not connect.

of the system as shown in Fig. 4.

#### IV. REAL-TIME LOCALIZATION FUNCTION BLOCKS

Fig. 4 presents the main function blocks of our VIRAL SLAM system. Most expansive of all is the localization thread, where all sensor data are synchronized and processed to eventually create factors in a cost function that is optimized using the ceres solver [27]. We will describe these processes in more details below.

##### A. Sensor data processing

1) *Lidar*: We refer to our previous work [15] for the details on the processing of lidar messages from multiple sensors. A quick recap is given below.

When each lidar pointcloud is obtained, feature extraction is immediately performed to produce a feature cloud. Hence, these feature clouds are synchronized and merged together to produce a single *skewed* combined feature cloud (SCFC), denoted as  $\tilde{\mathcal{F}}_m$ . The motion distortion (skew) in this SCFC is then further compensated by using the IMU-propagated states to obtain the body-relative deskewed combined feature cloud  ${}^B \mathcal{F}_m$ , which we refer to as CFC for short.

When the CFC  ${}^B \mathcal{F}_k$  is just obtained, we use the IMU-propagated state estimate at time  $t_k$ , denoted as  $\hat{\mathcal{X}}_k$ , to transform it to the local frame  $L$  while the previous  $M - 1$  CFCs, i.e.  ${}^B \mathcal{F}_m, m = w, w + 1, \dots, k - 1$ , are transformed to  $L$  using the state estimate  $\hat{\mathbf{T}}_m$  obtained from previous local optimizations. The set of  $M$   $L$ -referenced CFCs are then matched with the local map  $\mathcal{M}$  built from a subset of key frames in the memory. This subset is selected by searching for KNN of  $\hat{\mathcal{X}}_k$  among the key frame positions. The result of this feature-to-map matching (FMM) process is a set of coefficients, denoted as  $\{\mathcal{L}_m^i \triangleq (\mathbf{f}^i, \mathbf{n}^i, c^i)\}$ , whose cost factor is based on the following residual:

$$r_{\mathcal{L}}(\hat{\mathcal{X}}_m, \mathcal{L}_m^i) = (\mathbf{n}^i)^T [\hat{\mathbf{R}}_m \mathbf{f}^i + \hat{\mathbf{p}}_m] + c^i. \quad (6)$$

We note that the FMM process is one of the most time consuming tasks. However, it can be done on parallel threads [15], which can reduce the computation time proportionally with the number of CPU cores used in the system.

2) *IMU*: Given the IMU samples in the interval  $[t_{m-1}, t_m]$ , we can construct the preintegration observation  $\check{\mathcal{I}}_m \triangleq (\check{\alpha}_m, \check{\beta}_m, \check{\gamma}_m)$  that couples with two consecutive state  $\mathcal{X}_{m-1}$  and  $\mathcal{X}_m$ . Specifically,  $\check{\alpha}_m, \check{\beta}_m, \check{\gamma}_m$  are defined as

$$\check{\gamma}_m \triangleq \int_{s=t_{m-1}}^{t_m} \check{\gamma}_s \circ \begin{bmatrix} 0 \\ \check{\omega}_s - \hat{\mathbf{b}}_{m-1}^\omega \end{bmatrix}, \quad (7)$$

$$\check{\beta}_m \triangleq \int_{t_{m-1}}^{t_m} {}^{m-1}\check{\mathbf{R}}_s (\check{\mathbf{a}}_s - \hat{\mathbf{b}}_{m-1}^a) ds, \quad (8)$$

$$\check{\alpha}_m \triangleq \int_{t_{m-1}}^{t_m} \int_{t_{m-1}}^u {}^{m-1}\check{\mathbf{R}}_s (\check{\mathbf{a}}_s - \hat{\mathbf{b}}_{m-1}^a) ds du, \quad (9)$$

where  $\check{\mathbf{a}}_s$  and  $\check{\omega}_s$  are respectively the acceleration and angular velocity from IMU, and  $\check{\gamma}_{t_{m-1}} = 0$ ,  $\check{\beta}_{t_{m-1}} = 0$ ,  $\check{\alpha}_{t_{m-1}} = 0$ . The formulae for discrete time preintegration of IMU samples, as well as the residuals are given in our previous work [28].

Besides the preintegration, IMU data are also needed for propagating the state estimate from one time step to a later time instance for various purposes. For example, in Fig. 3, we need to propagate the state estimate  $\hat{\mathcal{X}}_{k-1}$  to  $t_k$  and  $t'_k$  to obtain the transform  ${}^L_B\check{\mathbf{T}}$  and  ${}^{t_k}\check{\mathbf{T}}$ . The latter is used for deskewing the SCFC  $\check{\mathcal{F}}_k$  to obtain the CFC  ${}^B\mathcal{F}_k$ , and the former is used to transform  ${}^B\mathcal{F}_k$  to  ${}^L\mathcal{F}_k$ . Moreover, the IMU propagated states are also used to find the relative transform  ${}_{\tau_k^i}^L\check{\mathbf{T}}$ , where  $\tau_k^i$  is the timestamp of the UWB sample. This transform is needed for the BA process in later part.

3) *UWB*: At the top of Fig. 4 is the workflow of the UWB measurements. From the *UWB ranging and communication* network, we can obtain the anchor-to-anchor and anchor-to-robot distance measurements. In the *anchor self-localization stage*, we take the mean of about 100 anchor-to-anchor measurements for each pair and use them as the prior  $r^{01}, r^{02}, r^{12}$  to calculate the coordinates of the anchors as described in Sec. III-B. Each robot-to-anchor range sample will be tagged with the corresponding anchor coordinate once the it has been set. In the *preliminary check* stage, we will check if the signal-over-noise and line of sight diagnoses of the UWB message are valid. After passing this check, the UWB samples will be synchronized with the time steps on the sliding window, as described in Sec. III-D. The measurement will finally be checked with IMU-propagated state to eliminate the outliers. The final results are the set of  $N_{\mathcal{U}}^m$  UWB measurements  $\{\mathcal{U}_m^i\}_{i=1}^{N_{\mathcal{U}}^m}$  for each interval  $(t_{m-1}, t_m]$ . Specifically,  $\mathcal{U}_m^i$  contains the following information:

$$\mathcal{U}_m^i = \left( \check{d}^i, {}^W\mathbf{x}^i, \mathbf{y}^i, \tau_m^i, t_{m-1}, t_m \right), \quad i = 1, 2, \dots, N_{\mathcal{U}}^m, \quad (10)$$

where  $\check{d}^i$  is the range measurement,  ${}^W\mathbf{x}^i$  is the coordinate of the anchor w.r.t.  ${}^W$ ,  $\mathbf{y}^i$  is the UAV ranging node in the body frame  ${}^B\tau_k^i$ ,  $\tau_m^i$  is the message's timestamp,  $t_{m-1}$  and  $t_m$  are the preceding and succeeding time steps of  $\tau_m^i$ .

We assume that the velocity and orientation of the robot change at a constant rate from time  $t_{m-1}$  to  $t_m$ . Thus, at time  $\tau_m^i$ , the relative position between the ranging node  $\mathbf{y}^i$  and the anchor  $\mathbf{x}^i$  can be determined as:

$$\begin{aligned} {}^L\mathbf{d}^i &= \mathbf{d}(\mathcal{X}_{m-1}, \mathcal{X}_m, {}^W\mathbf{T}, \mathcal{U}_m^i) \\ &\triangleq \mathbf{p}_m + \mathbf{R}_{m-1} \text{Exp}(s^i \text{Log}(\mathbf{R}_{m-1}^{-1} \mathbf{R}_m)) \mathbf{y}^i \\ &\quad - a^i \mathbf{v}_{m-1} - b^i \mathbf{v}_m - {}^W\mathbf{R} \mathbf{x}^i - {}^W\mathbf{t}, \end{aligned} \quad (11)$$

where  $s^i \triangleq \frac{\delta t_i}{\Delta t_m}$ ,  $a^i \triangleq \frac{\Delta t_m^2 - \delta t_i^2}{2\Delta t_m}$ ,  $b^i \triangleq \frac{(\Delta t_m - \delta t_i)^2}{2\Delta t_m}$ ,  $\delta t^i \triangleq \tau_k^i - t_{m-1}$ ,  $\Delta t_m \triangleq t_m - t_{m-1}$ .

We consider the distance measurement  $\check{d}^i$  at time  $t_k + \delta t^i$  as the norm of the vector  ${}^W\mathbf{d}^i$ , corrupted by a zero-mean Gaussian noise  $\eta_{\mathcal{U}^i} \sim \mathcal{N}(0, \sigma_{\mathcal{U}}^2)$  and the ranging bias introduced in (5), i.e.

$$\check{d}^i = \|{}^W\mathbf{d}^i\| + \eta_{\mathcal{U}^i} + \mathbf{b}^r. \quad (12)$$

Thus, the UWB range residual can be defined as:

$$\begin{aligned} r_{\mathcal{U}}(\hat{\mathcal{X}}_{m-1}, \hat{\mathcal{X}}_m, {}^W\hat{\mathbf{T}}, \hat{\mathbf{b}}^r, \mathcal{U}_m^i) \\ \triangleq \|\mathbf{d}(\hat{\mathcal{X}}_{m-1}, \hat{\mathcal{X}}_m, {}^W\hat{\mathbf{T}}, \mathcal{U}_m^i)\| + \hat{\mathbf{b}}^r - \check{d}^i. \end{aligned} \quad (13)$$

We note that the outlier check and fusion of UWB will not take place until the BA process has updated the anchor positions as well as the distance bias, which is discussed in Sec. V-C. Moreover, the state estimates  ${}^W\hat{\mathbf{T}}$  and  $\hat{\mathbf{b}}^r$  are fixed during the sliding window optimization process.

## B. Camera

Over the sliding window, BRIEF features are tracked across the images that are associated with the time steps as illustrated in Fig. 3. For a visual feature  $\mathbf{f}^i$  (Fig. 5) we define  $C_a$  as the first camera-attached frame of reference that observes it. Given the 3D coordinate  ${}^{C_a}\mathbf{f}^i \triangleq ({}^{C_a}x^i, {}^{C_a}y^i, {}^{C_a}z^i)$ , the corresponding *projected coordinates*  ${}^{C_a}\mathcal{Z}^i$ , and inverse depth  $\lambda^i$  associated with  $\mathbf{f}^i$  are defined as

$${}^{C_a}\mathcal{Z}^i \triangleq ({}^{C_a}\bar{x}^i, {}^{C_a}\bar{y}^i, 1) \quad (14)$$

$$\begin{aligned} &\triangleq \pi({}^{C_a}\mathbf{f}^i) \triangleq \frac{1}{{}^{C_a}z^i} ({}^{C_a}x^i, {}^{C_a}y^i, {}^{C_a}z^i) \\ &\triangleq \lambda^i ({}^{C_a}x^i, {}^{C_a}y^i, {}^{C_a}z^i) \end{aligned} \quad (15)$$

The projected coordinates of  $\mathbf{f}^i$  can be obtained from its pixel coordinates using the intrinsic parameters. After this conversion, features are triangulated by using the state estimates  $\hat{\mathcal{X}}_w, \hat{\mathcal{X}}_{w+1}, \dots, \hat{\mathcal{X}}_k$  to initialize the depth. Hence an initial guess of the 3D coordinates of the features can be obtained, denoted as  ${}^L\check{\mathbf{f}}^i$ . To further refine these results, we will search in the feature cloud  $\mathcal{M}$  for nearby lidar feature points. If at least a lidar point is within 0.25cm of  ${}^L\check{\mathbf{f}}^i$ , it can be used to construct a visual feature factor.

Given the projected coordinates of  $\mathbf{f}^i$  in another camera  $C_b$ , a visual residual can be constructed from a pair of

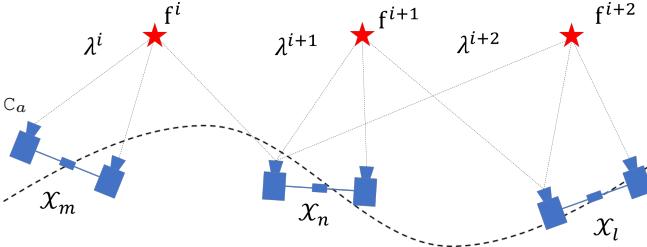


Fig. 5: Illustration of feature tracking scheme. Different from the lidar and UWB measurements, the indexing of visual factors are better based on the features. Here camera  $C_a$  that links to robot pose at time  $t_m$  is the first to detect a feature  $f^i$ , hence the inverse depth  $\lambda^i$  of feature  $f^i$  is associated with camera  $C_a$ .

observation  $\mathcal{V}_{ab}^i = (^{C_a}\mathcal{Z}^i, {}^{C_b}\mathcal{Z}^i)$  as follows:

$$\begin{aligned} r_{\mathcal{V}}(\hat{\mathcal{X}}_a, \hat{\mathcal{X}}_b, \mathcal{V}_{ab}^i) &= \pi\left({}^{C_b}\hat{\mathbf{f}}^i\right) - {}^{C_b}\mathcal{Z}^i, \\ {}^{C_b}\hat{\mathbf{f}}^i &\triangleq {}^{B_{C_b}}\mathbf{R}^{-1}\left({}^{m_b}\hat{\mathbf{f}}^i - {}^{B_{C_b}}\mathbf{t}\right), \\ {}^{m_b}\hat{\mathbf{f}}^i &\triangleq \hat{\mathbf{R}}_{m_b}^{-1}\left({}^L\hat{\mathbf{f}}^i - \hat{\mathbf{p}}_{m_b}\right), \\ {}^L\hat{\mathbf{f}}^i &\triangleq \hat{\mathbf{R}}_{m_a}{}^{m_a}\hat{\mathbf{f}}^i + \hat{\mathbf{p}}_{m_a}, \\ {}^{m_a}\hat{\mathbf{f}}^i &\triangleq {}^{B_{C_a}}\mathbf{R}\left((\hat{\lambda}^i)^{-1} {}^C_a\mathcal{Z}_i\right) + {}^{C_a}\mathbf{t}. \end{aligned} \quad (16)$$

Note that in this formulation,  ${}^{C_a}\mathbf{R}$ ,  ${}^{C_b}\mathbf{R}$ ,  ${}^{C_a}\mathbf{t}$ ,  ${}^{C_b}\mathbf{t}$  are constants extrinsic parameters of the cameras, and the frames  $C_a$  and  $C_b$  could be coupled with the same state but different cameras, or different states of the same camera, or both states and cameras are different.

### C. Sliding Window Optimization

After having processed all of the sensor data, we can combine all of the factors into the following cost function:

$$\begin{aligned} f(\hat{\mathcal{X}}) &\triangleq \left\{ \sum_{m=w+1}^k \left\| r_{\mathcal{I}}(\hat{\mathcal{X}}_{m-1}, \hat{\mathcal{X}}_m, \mathcal{I}_m) \right\|_{\mathbf{P}_{\mathcal{I}_m}^{-1}}^2 \right. \\ &+ \sum_{m=w}^k \sum_{i=1}^{N_{\mathcal{L}}^m} \rho_H \left( \left\| r_{\mathcal{L}}(\hat{\mathcal{X}}_m, \mathcal{L}_m^i) \right\|_{\mathbf{P}_{\mathcal{L}_m^i}^{-1}}^2 \right) \\ &+ \sum_{m=w}^k \sum_{i=1}^{N_{\mathcal{U}}^m} \left\| r_{\mathcal{U}}(\hat{\mathcal{X}}_{m-1}, \hat{\mathcal{X}}_m, {}^L\hat{\mathbf{T}}, \hat{\mathbf{b}}^r, \mathcal{U}_m^i) \right\|_{\mathbf{P}_{\mathcal{U}_m^i}^{-1}}^2 \\ &\left. + \sum_{i=1}^{N_{\mathcal{V}}^k} \sum_{b \in \mathcal{C}^i} \rho_A \left( \left\| r_{\mathcal{V}}(\hat{\mathcal{X}}_{m_a}, \hat{\mathcal{X}}_{m_b}, \hat{\lambda}^i, \mathcal{V}_{ab}^i) \right\|_{\mathbf{P}_{\mathcal{V}_{ab}^i}^{-1}}^2 \right) \right\}, \quad (17) \end{aligned}$$

where  $\rho_H(\cdot)$  and  $\rho_A(\cdot)$  are the Huber and arctan loss functions used to reduce the effects of outliers,  $N_{\mathcal{L}}^m \in \mathbb{N}$  is the number of FMM coefficients extracted from the CFC  $\mathcal{F}_m$ ,  $N_{\mathcal{U}}^k \in \mathbb{N}$  is the number of UWB samples obtained in the interval  $(t_{m-1}, t_m]$ ,  $N_{\mathcal{V}}^k \in \mathbb{N}$  is the number of visual features that are tracked on the sliding window from  $t_w$  to  $t_k$ , and  $\mathcal{C}^i$  refers to the set of cameras that observe the visual feature  $f^i$ , excluding  $C_a$ . The cost function (17) summarizes the coupling of each sensor's factor with the state estimate.

## V. GLOBAL OPTIMIZATION BLOCKS

### A. Key frame management

After each optimization on the local sliding window, we nominate all information associated with the middle time step, i.e.  $k - M/2$ , to be evaluated for marginalization as a key frame. Assuming  $(\hat{\mathbf{p}}_v, \hat{\mathbf{q}}_v)$ ,  $v = k - M/2$  is the pose of this key frame candidate, we will find its KNN among the existing key frame poses and create the sets  $\mathcal{N}_p = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_K\}$  and  $\mathcal{N}_q = \{\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_K\}$  that contain the positions and quaternions of these neighboring key frame poses. Then, we will check if the candidate's position is more than 0.5 m away, or its orientation is at least  $10^0$  away from the neighbors' poses, i.e.

- 1)  $\|\hat{\mathbf{p}}_v - \mathbf{p}\| > 0.5, \forall \mathbf{p} \in \mathcal{N}_p$ ,
- 2)  $\|\mathcal{E}^{-1}(\hat{\mathbf{q}}_v^{-1}\mathbf{q})\| > \pi/18, \forall \mathbf{q} \in \mathcal{N}_q$ .

If either one of the two conditions above is true, then we will admit the candidate as a new key frame. Upon its admission, its corresponding CFC  $\mathcal{F}_v$  is saved in the buffer for later use in the construction of local maps. In addition, we also save the visual features detected in the time step associated with this key frame for loop closure detection. Finally, all UWB samples whose timestamps are within 0.2 s to the keyframe timestamp's are also saved for use in the BA process.

### B. Loop Closure

The loop closure process is conducted through via the following three stages:

First, when a new key frame is added to the memory, we compare the set of visual features with the database using the DBow library. If the method returns a match, we use the index of the key frame to extract the transforms  $\mathbf{T}_c$  and  $\mathbf{T}_p$ , so-called current and previous key poses, respectively. Then, we search for KNN of  $\mathbf{T}_p$  and build a local map  ${}^{B_p}\mathcal{M}_p$  using their corresponding marginalized CFCs, and proceed to the second stage.

At the second stage, we will use ICP to align the CFC  ${}^{B_c}\mathcal{F}_c$  with  ${}^{B_p}\mathcal{M}_p$  to obtain a fitness score, as well as an initial guess of  ${}^{B_c}\hat{\mathbf{T}}$ . If the fitness score is below a threshold, meaning that the ICP process achieves a highly fit alignment, we proceed to the third stage.

At the third stage, we perform FMM between  ${}^{B_c}\mathcal{F}_c$  and  ${}^{B_p}\mathcal{M}_p$  to calculate the FMM coefficients, then construct the following cost function and optimize it:

$$f\left({}^{B_p}\hat{\mathbf{T}}\right) = \sum_{i=1}^{N_{\mathcal{L}}^c} \left\| \rho\left(r_{\mathcal{L}}({}^{B_c}\hat{\mathbf{T}}, \mathcal{L}_c^i)\right) \right\|_{\mathbf{P}_{\mathcal{L}_c^i}^{-1}}^2. \quad (18)$$

After optimizing (18) and obtain the optimal relative pose  ${}^{B_p}\hat{\mathbf{T}}^*$ , if the ratio  $f\left({}^{B_p}\hat{\mathbf{T}}^*\right)/N_{\mathcal{L}}^c$  is below a threshold, we will admit  ${}^{B_p}\hat{\mathbf{T}}^*$  as a loop closure prior  ${}^{B_p}\mathbf{T}^*$  to the BA process.

### C. Bundle Adjustment

In the BA process, the main task is to optimize a pose graph whose cost function can be described as:

$$f(\hat{\mathcal{Y}}, {}^L_W \hat{\mathbf{T}}, \hat{\mathbf{b}}^r) \triangleq \left\{ \sum_{n=1}^N \left\| \mathbf{r}_1(\hat{\mathbf{T}}_{n-1}, \hat{\mathbf{T}}_n, {}^{n-1} \mathbf{T}^*) \right\|_{\mathbf{P}_1^{-1}}^2 + \sum_{(p,c) \in \mathcal{H}} \left\| \mathbf{r}_2(\hat{\mathbf{T}}_p, \hat{\mathbf{T}}_c, {}^p \mathbf{T}^*) \right\|_{\mathbf{P}_3^{-1}}^2 + \sum_{n=1}^N \sum_{j=1}^{N_U^n} \left\| \mathbf{r}_3(\hat{\mathbf{T}}_n, {}^L_W \hat{\mathbf{T}}, \hat{\mathbf{b}}^r, \mathcal{U}_n^{*i}) \right\|_{\mathbf{P}_2^{-1}}^2 \right\}, \quad (19)$$

where  $\hat{\mathcal{Y}} = (\hat{\mathbf{T}}_0, \hat{\mathbf{T}}_1, \dots, \hat{\mathbf{T}}_N)$ ,  ${}^{n-1} \mathbf{T}^*$  and  ${}^p \mathbf{T}^*$  are respectively the relative pose and loop closure priors,  $\mathcal{H}$  is the set of loop closure index pairs, and  $\mathcal{U}_n^{*i}$  is a UWB measurement whose timestamp is within 0.2s of the key frame at time  $t_n$ . Specifically,  $\mathcal{U}_n^{*i}$  consists of the following information:

$$\mathcal{U}_n^{*i} = \left( \breve{d}^i, {}^W \mathbf{x}^i, \mathbf{y}^i, {}^{t_n} \mathbf{R}^*, {}^{t_n} \mathbf{t}^* \right), \quad (20)$$

where  $\breve{d}^i$ ,  ${}^W \mathbf{x}^i$ ,  $\mathbf{y}^i$  are defined similarly to (10), and  $({}^{t_n} \mathbf{R}^*, {}^{t_n} \mathbf{t}^*)$  is the relative transform between  $B_{t_n}$  and  $B_{\tau_n^i}$ , which can be obtained from IMU propagation. After each optimization steps, the relative pose priors are also updated with the optimized results.

The residuals  $\mathbf{r}_1(\cdot)$ ,  $\mathbf{r}_2(\cdot)$  over the relative poses are straightforward, while the residual  $\mathbf{r}_3$  can be stated as follows:

$$\mathbf{r}_3(\hat{\mathbf{T}}_n, {}^L_W \hat{\mathbf{T}}, \hat{\mathbf{b}}^r, \mathcal{U}_n^{*i}) = \left\| \hat{\mathbf{d}}^i \right\| + \hat{\mathbf{b}}^r - \breve{d}^i, \\ \hat{\mathbf{d}}^i \triangleq \hat{\mathbf{p}}_n + \hat{\mathbf{R}}_n {}^{t_n} \mathbf{R}^* (\mathbf{y}^i + {}^{t_n} \mathbf{t}^*) - {}^L_W \hat{\mathbf{R}} \mathbf{x}^i - {}^L_W \hat{\mathbf{t}}. \quad (21)$$

To ensure the pose graph has enough excitation to help the the anchor-related states converge, we do not add the factors of  $\mathbf{r}_3$  into the BA cost function right from the beginning. Rather, they are only added in after the spatial distribution of the key frame poses has satisfied a certain condition. Specifically, we calculate the geometric dilution of the key frame positions via the quantity

$$\Gamma = \left( \sum_{n=1}^N (\mathbf{p}_n^* - \mu)(\mathbf{p}_n^* - \mu)^\top \right)^{-1}, \quad \mu = \frac{1}{N} \sum_{n=1}^N \mathbf{p}_n^*. \quad (22)$$

Hence we perform singular value decomposition on  $\Gamma$  to obtain the singular values  $\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0$ . If  $\sigma_1 < c_1$  and  $\sigma_1/\sigma_3 < c_2$ , where  $c_1$  and  $c_2 > 1$  are some user-defined parameters, then we can start adding the range factors  $\mathbf{r}_3(\cdot)$  to (19). After this BA result, we can obtain the estimates of  ${}^L_W \mathbf{T}$  and  $\mathbf{b}^r$  which can be used for the fusion of UWB factors in (17).

## VI. EXPERIMENT

In this part we present the experiments to demonstrate the capability of VIRAL SLAM. Video recording of some of our works can be viewed at <https://youtu.be/6s7CcydJLfg>.

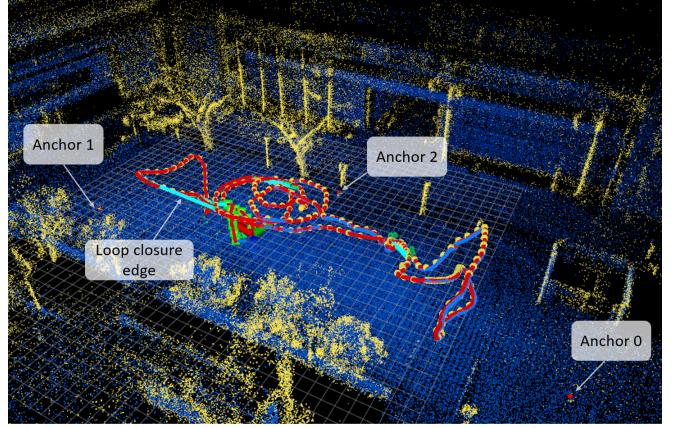


Fig. 6: VIRAL SLAM result on eee\_02 dataset. The estimated trajectory is in blue, and ground truth is in red. The key frame poses are marked by the yellow circles. The activated key frames for local map building are highlighted by the green squares. The UWB anchors are also marked with red dots. The loop edges are marked with light cyan lines. Some visual features can be marked with small green circles.

### A. NTU VIRAL Datasets

#### B. Comparison with other works

We first employ our recently published NTU VIRAL dataset<sup>1</sup> [28], which features all sensor types covered by VIRAL SLAM, to evaluate the method and compare it with other state of the art localization techniques. Specifically, VINS-Mono<sup>2</sup> [11], VINS-Fusion<sup>3</sup>, A-LOAM<sup>4</sup>, LIO-SAM<sup>5</sup>, M-LOAM<sup>6</sup> [14] are considered. We also attempted to run LVI-SAM and ORB-SLAM3 with the dataset. However the LVI-SAM quickly diverges after the drone takes off, while ORB-SLAM3 loses track quite easily. All algorithms are run on an NUC 10 computer with core i7 processor. Each method is slightly modified and configured for their best performance with the dataset. The details of these modified packages can be found on NTU VIRAL dataset website<sup>1</sup>. Since several lidar-based methods are not designed to incorporate multiple lidar inputs, we also include experiments of VIRAL SLAM using only the horizontal lidar for a fairer comparison.

Table I presents the positional root-mean-square error (RMSE) of VIRAL SLAM and other methods. From this table we can clearly see that VIRAL SLAM consistently achieves better performance compared to existing methods, even when only one lidar is used. For VINS-Mono, VINS-Fusion and VIRAL SLAM, we also report the BA results, i.e. the positions of the key frames refined by loop closure and BA process. We note that the pure odometry and BA results of VIRAL SLAM do not differ much, while there is a large difference between odometry and BA results of the

<sup>1</sup>[https://ntu-aris.github.io/ntu\\_viral\\_dataset/](https://ntu-aris.github.io/ntu_viral_dataset/)

<sup>2</sup><https://github.com/brytsknguyen/VINS-Mono>

<sup>3</sup><https://github.com/brytsknguyen/VINS-Fusion>

<sup>4</sup><https://github.com/brytsknguyen/A-LOAM>

<sup>5</sup><https://github.com/brytsknguyen/LIO-SAM>

<sup>6</sup><https://github.com/brytsknguyen/M-LOAM>

TABLE I: ATE of the localization methods over the NTU VIRAL datasets. The best odometry result is highlighted in **bold**, and the second best is underlined. All values are in m.

Dataset	VINS-Mono (right camera, odom/BA)	VINS-Fusion (both cameras, odom/BA)	A-LOAM (horz. / vert.)	LIO- SAM (horz.)	MLOAM (2 lidars)	VIRAL-SLAM (horz. lidar, odom/BA)	VIRAL-SLAM (2 lidars, odom/BA)
eee_01	1.651 / 0.569	0.586 / 0.327	0.224 / 5.713	0.091	0.380	<b>0.083</b> / 0.086	<u>0.089</u> / 0.088
eee_02	0.747 / 0.469	0.575 / 0.217	0.189 / 2.244	<b>0.080</b>	0.202	0.084 / 0.064	<u>0.084</u> / 0.059
eee_03	1.037 / 0.886	0.582 / 0.467	0.152 / 4.380	0.118	0.278	<u>0.057</u> / 0.067	<b>0.045</b> / 0.057
nya_01	1.775 / 1.090	0.398 / 0.234	0.080 / 0.803	0.090	0.145	<u>0.060</u> / 0.060	<b>0.045</b> / 0.059
nya_02	0.580 / 0.481	0.424 / 0.298	0.093 / 0.378	0.107	0.235	<b>0.061</b> / 0.066	<u>0.069</u> / 0.058
nya_03	1.333 / 0.514	0.786 / 0.390	0.084 / 0.683	0.360	0.283	0.053 / 0.076	<b>0.037</b> / 0.069
sbs_01	4.081 / 3.686	0.507 / 0.372	0.195 / 6.452	0.094	0.188	<b>0.068</b> / 0.061	<u>0.070</u> / 0.051
sbs_02	1.607 / 0.891	0.563 / 0.369	0.087 / 2.816	0.096	0.177	<b>0.068</b> / 0.067	<u>0.078</u> / 0.058
sbs_03	1.306 / 0.802	0.841 / 0.295	0.346 / 4.732	0.103	0.182	<u>0.071</u> / 0.068	<b>0.070</b> / 0.077

TABLE II: RMSE of VIRAL SLAM’s key frame positions with different combinations of sensors over the NTU VIRAL datasets (IMU is always used). All values are in m.

Dataset	Lidars	Lidars +Cameras	Lidars +UWB	Lidars +UWB +Cameras
eee_01	<b>0.0552</b>	<u>0.0557</u>	0.0867	0.0878
eee_02	0.0598	<u>0.0594</u>	0.0602	<b>0.0587</b>
eee_03	0.0660	0.0666	<u>0.0572</u>	<b>0.0567</b>
nya_01	<u>0.0492</u>	<b>0.0491</b>	0.0618	0.0591
nya_02	<u>0.0536</u>	<b>0.0526</b>	0.0752	0.0581
nya_03	<b>0.0464</b>	0.0483	0.0749	0.0692
sbs_01	0.0685	0.0647	<u>0.0585</u>	<b>0.0507</b>
sbs_02	0.0639	0.0643	<u>0.0579</u>	<b>0.0579</b>
sbs_03	0.070 0	<u>0.0694</u>	<b>0.0653</b>	0.0767
Average	<u>0.0592</u>	<b>0.0589</b>	0.0664	0.0639

VIO methods. This can be explained as VIRAL SLAM has much less drift than VINS methods, thus there is not a lot of correction made in the BA process. Fig. 6 presents the result of VIRAL SLAM in the second experiment.

1) *Ablation study*: We further conduct extra experiments with different subsets of sensor suites to study the contribution of each sensor in the overall localization scheme. Tab. II reports the RMSE of the key frame positions of VIRAL SLAM over the NTU VIRAL datasets. From this table we can see that the lidar-only and lidar-camera setups has roughly similar average RMSE over the datasets. From lidar-only to lidar-range, there is a significant increase in error. This can be explained as with UWB we are not just estimating the key frame poses, but also states relating to the UWB anchor position and bias. Thus the increase in RMSE is a trade-off of knowledge on the UWB states. Interestingly, when comparing the RMSE of lidar-range setup and lidar-range-camera setup, we can observe a decrease in RMSE. Thus, we can conclude that the loop closure constraints introduced by using camera do help bring down the error in the BA process.

### C. Building inspection scenario

1) *Comparison with other works*: The NTU VIRAL dataset is helpful showing the accuracy of our approach compared to other methods. However, the testing condi-

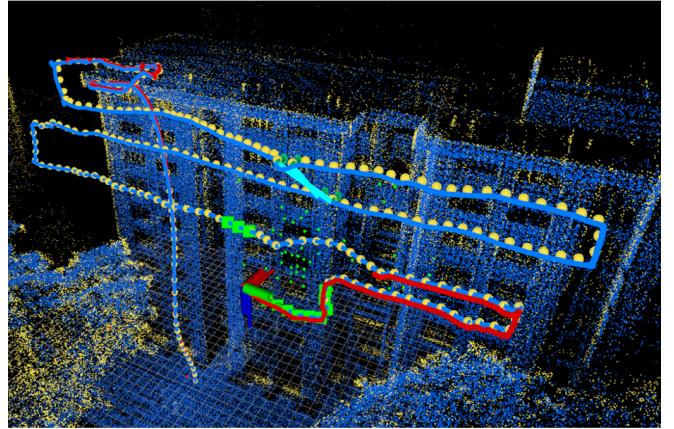


Fig. 7: VIRAL SLAM on blk\_02 dataset. The color scheme is similar to to Fig. 6.

tions of these datasets are relatively ideal where are plenty of features by the surrounding building for cameras and lidars. Thus a single lidar can already achieve relatively good localization results. However, besides accuracy, the combination of several sensors together also aims to achieve robustness, such that in critical conditions localization can still be maintained. To bring out this capability of VIRAL SLAM, we conduct further experiments on some datasets with significant challenges collected near a building facade.

Specifically, for lidar, there are only the vertical facade and the horizontal ground plane that can be observed (see Fig. 7), but each lidar can only effective scan one of these two planes. Hence, the horizontal lidar will have difficulty correcting drift in the vertical direction, while the vertical lidar will drift in the lateral direction. Moreover, visual features are mostly located on the facade plane and the scenes are most repetitive, posing a challenge to the loop closure process.

The same localization methods are run with this dataset and results are reported in Table III. On the first look, we can recognize that the accuracy of all methods have increased significantly compared to with the NTU VIRAL dataset. LIO-SAM fails to work as its front-end fails to track the features. Even for VIRAL SLAM with only one lidar, there is still one experiment where substantial drift

TABLE III: RMSE of the SLAM methods over the building inspection datasets. The best odometry result is highlighted in **bold**, and the second best is underlined. All values are in m.

Dataset	VINS-Mono (right camera, odom/BA)	VINS-Fusion (both cameras, odom/BA)	A-LOAM (horz. / vert.)	LIO-SAM (horz.)	MLOAM (2 lidars)	VIRAL-SLAM (horz. lidar, odom/BA)	VIRAL-SLAM (2 lidars, odom/BA)
bid.01	3.731 / 3.613	2.416 / 2.077	1.022 / 12.844	-	4.265	0.180 / 0.158	<b>0.163</b> / 0.157
bid.02	1.256 / 1.241	0.837 / 0.625	5.713 / 4.317	-	<b>0.267</b>	0.564 / 0.946	0.343 / 0.607
bid.03	0.670 / 0.659	0.925 / 0.824	3.084 / 1.682	-	3.331	1.626 / 2.953	<b>0.129</b> / 0.177

TABLE IV: RMSE of VIRAL SLAM’s key frame positions with different sensor combinations over the building inspection datasets (IMU is always used). All values are in m.

Dataset	Lidars	Lidars +Cameras	Lidars +UWB	Lidars +UWB +Cameras
bid.01	0.2355	0.2510	<u>0.1574</u>	<b>0.1568</b>
bid.02	<b>0.6000</b>	<u>0.6035</u>	0.6119	0.6068
bid.03	0.1943	0.1917	<u>0.1783</u>	<b>0.1769</b>
Average	0.3433	0.3487	<u>0.3159</u>	<b>0.3135</b>

occurs. Nevertheless, VIRAL-SLAM utilizing both lidars still achieve good performance compared to others. Fig. 7 shows the trajectory estimate of VIRAL SLAM in one of these experiments.

2) *Ablation study*: Table IV shows the results of using different combinations of sensors with the building inspection datasets. We can see that in this challenging environment, there is significant improvement when using either camera or UWB, or both camera and UWB, to supplement the lidar capability.

#### D. AirSim Datasets

1) *Comparison with other methods*: In the previous datasets, we have estimated the position of the anchors using the self-localization method described in Sec. III-B, which are refined via the BA process. However no ground truth on the anchor position, as well as the ranging bias are available. To clearly verify this capability of VIRAL-SLAM, we employ AirSim simulator to construct a dataset with absolute ground truth for more accurate evaluate.

The NTU VIRAL dataset is helpful showing the accuracy of our approach compared to other methods. However, the testing conditions of these datasets are relatively ideal where are plenty of features by the surrounding building for cameras and lidars. Thus a single lidar can already achieve relatively good localization results. However, besides accuracy, the combination of several sensors together also aims to achieve robustness, such that in critical conditions localization can still be maintained. To bring out this capability of VIRAL SLAM, we conduct further experiments on some datasets with significant challenges collected near a building facade.

Specifically, for lidar, there are only the vertical facade and the horizontal ground plane that can be observed (see Fig. 7), but each lidar can only effective scan one of these two planes. Hence, the horizontal lidar will have difficulty correcting

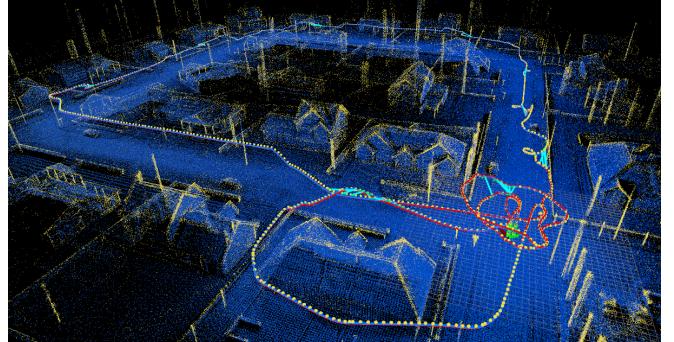


Fig. 8: Trajectory of the UAV as estimated by VIRAL SLAM and ground truth in an experiment over the *Neighborhood* environment of AirSim. Note that there are several loops created intentionally over the trajectory by rotating the UAV.

drift in the vertical direction, while the vertical lidar will drift in the lateral direction. Moreover, visual features are mostly located on the facade plane and the scenes are most repetitive, posing a challenge to the loop closure process.

The same localization methods are run with this dataset and results are reported in Table III. On the first look, we can recognize that the accuracy of all methods have increased significantly compared to with the NTU VIRAL dataset. LIO-SAM even fails to work as its front-end fails to track the features. Even for VIRAL SLAM with only one lidar, there is still one experiment where substantial drift occurs. Nevertheless, VIRAL-SLAM utilizing both lidars still achieve good performance compared to others. Fig. 7 shows the trajectory estimate of VIRAL SLAM in one of these experiments.

Table V again confirms the advantage of VIRAL-SLAM compared with other existing methods, as they have the best odometry results among all other methods. Fig. 8 shows the result of one experiment.

#### E. Ablation study

We again conduct the ablation study to understand the effects the sensors in the SLAM results. From Table VII, we can see a similar pattern as in the case with the NTU VIRAL datasets, where lidar-only and lidar-camera setups achieve similar RMSE, while the lidar-range setup sees an increase in RMSE, and decrease when camera is added to the system. Different from the NTU VIRAL dataset, the increase in RMSE from lidar-only to the lidar-range appears to be more significant. This can be explained as

TABLE V: RMSE of the SLAM methods over the AirSim-generated datasets. The best odometry result is highlighted in **bold**, and the second best is underlined.

Dataset	VINS-Mono (right camera, odom/BA)	VINS-Fusion (both cameras, odom/BA)	A-LOAM (horz. / vert. / latr.)	LIO- SAM (horz.)	MILOAM (3 lidars)	VIRAL-SLAM (horz. lidar, odom/BA)	VIRAL-SLAM (3 lidars, odom/BA)
nbh_01	4.761 / 4.517	1.414 / 1.387	86.476 / 58.160	-	0.474	<b>0.179</b> / 0.178	0.188 / 0.242
nbh_02	3.527 / 3.019	2.351 / 1.162	45.965 / 42.150	-	0.538	<b>0.103</b> / 0.188	<u>0.115</u> / 0.152
nbh_03	3.559 / 2.748	1.377 / 0.933	7.233 / 50.612	-	0.340	<u>0.177</u> / 0.199	<b>0.167</b> / 0.137
nbh_04	2.706 / 1.986	1.974 / 1.512	25.014 / 36.498	-	0.516	<u>0.127</u> / 0.183	<b>0.114</b> / 0.160
nbh_05	119.117 / 121.639	1.255 / 0.814	0.836 / 25.672	-	0.544	<b>0.146</b> / 0.254	<u>0.152</u> / 0.303

TABLE VII: RMSE of VIRAL SLAM's key frame positions with different sensor combinations over the AirSim datasets (IMU is always used). All values are in m.

Dataset	Lidars	Lidars +Cameras	Lidars +UWB	Lidars +UWB +Cameras
nbh_01	<b>0.0462</b>	<u>0.0480</u>	0.2431	0.2424
nbh_02	<b>0.0469</b>	<u>0.0560</u>	0.1598	0.1517
nbh_03	<b>0.0362</b>	<u>0.0386</u>	0.1568	0.1373
nbh_04	<b>0.0221</b>	<u>0.0305</u>	0.1726	0.1598
nbh_05	<b>0.0329</b>	<u>0.0336</u>	0.3009	0.3026
Average	<b>0.0369</b>	<u>0.0413</u>	0.2066	0.1988

TABLE VI: Anchor coordinates and the final values estimated by the BA process. All values are in m. Note that the coordinates of anchor 0 is fixed at (0, 0, 1).

	Anchor 1	Anchor 2	$b^r$
<b>True values</b>	15.000, 0.000, 1.250	7.500, -5.000, 1.500	0.050
<b>Initial values</b>	15.020, 0.000, 1.000	7.450, -5.070, 1.000	0.000
nbh_01 est.	15.017, 0.080, 1.275	7.468, -5.014, 1.547	0.032
nbh_02 est.	15.017, -0.020, 1.299	7.434, -5.064, 1.554	0.014
nbh_03 est.	15.014, 0.043, 1.426	7.450, -5.032, 1.622	0.029
nbh_04 est.	15.015, 0.028, 1.402	7.446, -5.041, 1.594	0.038
nbh_05 est.	15.019, 0.015, 1.191	7.450, -5.052, 1.421	0.018

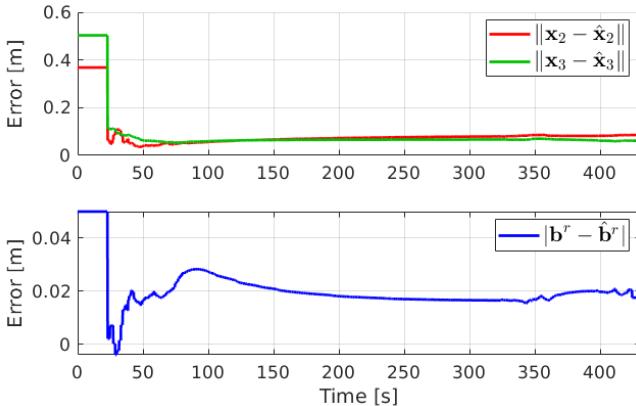


Fig. 9: Error of the estimates on anchor position and ranging bias by the BA process over time.

UWB measurements are mostly only available for the key frames near the starting point. Thus, the solution from the BA process can be unbalanced where the key frame positions near the starting point seem to fit the ground truth trajectory better than those farther away.

## VII. CONCLUSION AND FUTURE WORKS

In this paper we have developed a multi-sensor SLAM method leveraging an extensive set of sensors: stereo camera, lidar, IMU, UWB, so-called VIRAL SLAM. The system features synchronization and integration of multiple lidars with complementary field of view (FOV), depth-matching of stereo camera visual with pointcloud map, loop closure detection, UWB extrinsic and intrinsic parameter estimation. Via extensive experiments results, we have demonstrated that VIRAL SLAM can achieve highly accurate localization result as well as robustness in challenging conditions.

## REFERENCES

- J. Zhang and S. Singh, "Laser-visual-inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.
- T. Shan, B. Englot, C. Ratti, and R. Daniela, "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. to-be-added.
- T.-M. Nguyen, M. Cao, S. Yuan, Y. Lyu, T. H. Nguyen, and L. Xie, "Liro: Tightly coupled lidar-inertia-ranging odometry," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Accepted, 2020.
- , "Viral-fusion: A visual-inertial-ranging-lidar sensor fusion approach," *Submitted to IEEE Transactions on Robotics*, 2021.
- T. H. Nguyen, T.-M. Nguyen, and L. Xie, "Tightly-coupled ultra-wideband-aided monocular visual slam with degenerate anchor configurations," *Autonomous Robots*, vol. 44, no. 8, pp. 1519–1534, 2020.
- , "Range-focused fusion of camera-imu-uwb for accurate and drift-reduced localization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1678 – 1685, 2021.
- X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "Lic-fusion: Lidar-inertial-camera odometry," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5848–5854.
- X. Zuo, Y. Yang, P. Geneva, L. Jiajun, Y. Liu, G. Huang, and M. Pollefey, "Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5112–5119.
- J. Graeter, A. Wilczynski, and M. Lauer, "Limo: Lidar-monocular visual odometry," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 7872–7879.
- W. Shao, S. Vijayarangan, C. Li, and G. Kantor, "Stereo visual inertial lidar simultaneous localization and mapping," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2019.
- T. Qin, 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.
- T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

- [13] T. Shan and B. Englot, “Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.
- [14] J. Jiao, H. Ye, Y. Zhu, and M. Liu, “Robust odometry and mapping for multi-lidar systems with online extrinsic calibration,” *arXiv preprint arXiv:2010.14294*, 2020.
- [15] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, “Miliom: Tightly coupled multi-input lidar-inertia odometry and mapping,” *Submitted to IEEE Robotics and Automation Letters*, 2021.
- [16] M. Karrer and M. Chli, “Distributed variable-baseline stereo slam from two uavs,” *arXiv preprint arXiv:2009.04801*, 2020.
- [17] T.-M. Nguyen, T. H. Nguyen, M. Cao, Z. Qiu, and L. Xie, “Integrated uwb-vision approach for autonomous docking of uavs in gps-denied environments,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9603–9609.
- [18] T.-M. Nguyen, Z. Qiu, T. H. Nguyen, M. Cao, and L. Xie, “Distance-based cooperative relative localization for leader-following control of mavs,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3641–3648, 2019.
- [19] ——, “Persistently excited adaptive relative localization and time-varying formation of robot swarms,” *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 553–560, 2019.
- [20] J. P. Queralta, L. Qingqing, F. Schiano, and T. Westerlund, “Vio-uwb-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems,” *arXiv preprint arXiv:2011.00830*, 2020.
- [21] J. Xu, J. Hu, L. Xie, and K.-Y. Lum, “Distributed coverage control under generalized locational optimization framework,” in *Proceedings of the 31st Chinese Control Conference*. IEEE, 2012, pp. 6015–6020.
- [22] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, “Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8776–8782.
- [23] H. Ye, Y. Chen, and M. Liu, “Tightly coupled 3d lidar inertial odometry and mapping,” in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.
- [24] C. Forster, L. Carbone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual-inertial odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.
- [25] D. Gálvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.
- [26] T.-M. Nguyen, A. H. Zaini, C. Wang, K. Guo, and L. Xie, “Robust target-relative localization with ultra-wideband ranging and communication,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2312–2319.
- [27] S. Agarwal, K. Mierle, and Others, “Ceres solver,” <http://ceres-solver.org>.
- [28] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, “Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint,” *Submitted to IJRR*, 2021.