UNIVERSITY OF CALIFORNIA
RIVERSIDE

Towards AI-Aided Multi-User AR: Cooperative Visual-Inertial Odometry Enhanced by Point-Line Features and Neural Radiance Fields

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Yanyu Zhang

March 2025

Dissertation Committee:

Dr. Wei Ren, Chairperson
Dr. Amit K. Roy-Chowdhury
Dr. Hyoseung Kim

The Dissertation of Yanyu Zhang is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to my advisor, Dr. Wei Ren. This journey would not have been possible without his unwavering guidance, support, and encouragement over the past four years. His dedication, expertise, and passion for research have been a profound source of inspiration for me. He has provided me with invaluable advice, the necessary tools and resources, and the flexibility to accomplish my studies and succeed in my work. I feel extremely lucky to be his student. The skills and knowledge I have gained under his mentorship will undoubtedly serve me well in my future career.

I extend my thanks to the faculty members who have offered guidance and support throughout my studies. I am particularly grateful to Dr. Hang Qiu, Dr. Srikanth V. Krishnamurthy, Dr. Jiasi Chen, Dr. Matt Barth, and Dr. Guoyuan Wu for their insightful suggestions and advice on my research. The discussions with them have greatly enriched my work. I would also like to thank the members of my oral and defense committee, Dr. Amit K. Roy-Chowdhury, Dr. Hyoseung Kim, Dr. Jay A. Farrell, Dr. Ran Cheng, and Dr. Ahmed Eldawy, for their valuable feedback and constructive comments, which have significantly enhanced the quality of this dissertation.

I am deeply appreciative of the financial support provided by the National Science Foundation (Grant no. CMMI-2027139) and the Department of Electrical and Computer Engineering at UCR. I also owe thanks to my amazing colleagues in the COVEN Lab, Jie Xu, Dongming Wang, Yuhan Zhu, Shaoshu Su, Dr. Pengxiang Zhu, Dr. Yong Ding, Dr. Shan Sun, as well as the visiting scholars. Your camaraderie, encouragement, and invaluable

discussions have made this journey more enjoyable and enriching. Meanwhile, I would also like to express my gratitude to my friends, Bo Wu, Haishan Liu, Haoge Zhou, Wenying Wu, Xiaolin Luo, and Xingyu Lu, for bringing color and joy to my graduate life. Your companionship has made this experience unforgettable, and I wish you all the best in your future endeavors.

Finally, I owe my deepest gratitude to my family. I am forever indebted to my parents, Jie Zhang and Haijuan Li, for their boundless love and care. To my wife, Mengyuan Liu, thank you for standing by my side throughout this journey. Your unwavering support, understanding, and encouragement have meant the world to me, and I am forever grateful.

Dedicated to my parents,

Jie Zhang and Haijuan Li,

and to my wife, Mengyuan Liu.

Your love lights my way and gives me strength.

ABSTRACT OF THE DISSERTATION

Towards AI-Aided Multi-User AR: Cooperative Visual-Inertial Odometry Enhanced by Point-Line Features and Neural Radiance Fields

by

Yanyu Zhang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, March 2025
Dr. Wei Ren, Chairperson

This dissertation presents a suite of novel methodologies designed to advance multi-user augmented reality (AR) systems by addressing challenges in localization, mapping, and real-time collaboration. Key contributions focus on enhancing visual-inertial odometry (VIO) and introducing infrastructure-less cooperative SLAM techniques.

Firstly, a Point-Line Cooperative Visual-Inertial Odometry (PL-CVIO) framework is proposed to improve localization accuracy, particularly in low-feature environments. By integrating point and line features and enabling feature sharing between neighboring robots, PL-CVIO leverages geometric constraints to achieve robust, cooperative localization. The framework employs covariance intersection (CI) to ensure consistent state estimation across multiple agents.

Secondly, a novel map-assisted VIO system is introduced by leveraging Neural Radiance Fields (NeRF) to encode compact and photorealistic 3D maps. These maps provide robust geometric constraints for localization, addressing key challenges such as pose initialization, drift correction, and environmental adaptability. A pose initialization model

is proposed by using geodesic errors. Besides, an online VIO algorithm is developed, which leverages both real-world and NeRF-rendered images to update the state, demonstrating significant improvements in accuracy and robustness.

Thirdly, we propose CooperSLAM, a lightweight, infrastructure-free cooperative SLAM algorithm designed for multi-user AR in dynamic and resource-limited environments. CooperSLAM enables efficient peer-to-peer communication and sparse map feature sharing, enhancing scalability while reducing bandwidth requirements. By decoupling map points and key frames and introducing opportunistic relocalization strategies, CooperSLAM facilitates effective collaboration without reliance on centralized infrastructure.

Extensive simulations and real-world experiments validate the performance of the proposed methods. Results demonstrate substantial improvements in localization accuracy, robustness, and scalability compared to existing methods. This work contributes to the development of intelligent, collaborative AR systems designed to function effectively in dynamic and infrastructure-less environments, offering potential applications in immersive technologies, robotics, and related fields.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the age of robotics and immersive technologies, precise localization, efficient navigation, and robust sensor integration are the pillars of simultaneous localization and mapping (SLAM), autonomous driving (AD), and augmented/virtual reality (AR/VR). These systems rely on the seamless collaboration of multiple sensing modalities and agents, with applications spanning autonomous driving, multi-agent exploration, disaster recovery, and interactive AR. However, significant challenges persist, including infrastructure dependence, high computational overhead, sensor synchronization, and robustness in complex environments.

In GPS-denied environments, high-precision localization becomes essential for reliable operation. Visual-inertial navigation systems (VINS), which combine cameras and inertial sensors, have emerged as lightweight and cost-effective solutions. However, traditional VINS approaches struggle with accumulated drift, reliance on dense environmental

features, and the inability to leverage multi-agent collaboration effectively. This is especially problematic in human-made or low-feature environments, where traditional feature-based methods face significant limitations.

Neural radiance fields (NeRF) provide a breakthrough in addressing these challenges by encoding 3D scene geometry and appearance into compact representations. Unlike conventional maps based on sparse features or dense point clouds, NeRF employs a volumetric representation through a multi-layer perceptron, enabling photorealistic rendering of any viewpoint. This capability facilitates the creation of 3D geometric and semantic maps, which enhance visual-inertial odometry (VIO) systems by providing strong geometric constraints between the observed environment and the encoded prior map. However, integrating NeRF into VIO systems introduces challenges such as achieving real-time performance and accurately initializing the pose relative to the map.

Additionally, AR applications increasingly demand real-time, multi-user collaboration in shared environments. Existing solutions often rely on cloud-based infrastructure for localization and mapping, which introduces latency, high bandwidth requirements, and dependency on stable network connectivity. These limitations restrict the scalability and applicability of AR technologies in scenarios like disaster recovery, remote exploration, and infrastructure-free environments.

1.2 Contributions

1.2.1 Point-Line Cooperative Visual-Inertial Odometry

Low-feature environments pose a significant challenge to traditional localization and mapping systems. In such scenarios, geometric constraints derived from point and line features offer a complementary solution. This thesis introduces PL-CVIO, a cooperative visual-inertial odometry algorithm that integrates point-line features with a distributed multi-robot framework. The proposed method employs the multi-state constraint Kalman filter (MSCKF) architecture, utilizing left-nullspace projection for processing independent features and the covariance intersection (CI) for updating common features observed by multiple robots. Extensive simulations and real-world experiments validate the effectiveness of PL-CVIO, demonstrating substantial improvements in localization accuracy and robustness, particularly in low-feature environments.

1.2.2 Map-Based Visual-Inertial Odometry Leveraging NeRF

We also explores the integration of neural representations to enhance VIO. NeRF provides a transformative approach to encoding environmental information into compact, photorealistic representations, facilitating the creation of prior maps that significantly improve localization accuracy. However, current NeRF-based methods often encounter challenges with real-time performance and robust initialization. To address these issues, we develop a novel pose estimation model to initialize the first inertial measurement unit (IMU) state of a VINS system within the prior map frame. This model leverages a multi-layer perceptron (MLP) to encode map-pose relationships and introduces a novel loss function

based on geodesic errors on $SE(3)$. Besides, we prove the left-invariant of our proposed loss function. Additionally, we propose an online NeRF-based VIO algorithm (NeRF-VIO) that integrates the NeRF-based prior map with the initialization model. By utilizing both real-world images captured from an onboard camera and rendered images from the NeRF model, NeRF-VIO performs state updates efficiently and accurately. Finally, the effectiveness of the proposed method is validated using a real-world AR dataset.

1.2.3 Infrastructure-less Cooperative SLAM for Multi-user AR

We address the limitations of cloud-based infrastructure-dependent solutions for multi-user AR and cooperative mapping, which restrict deployment in remote or infrastructure-less environments. To overcome these challenges, this work introduces CooperSLAM, a lightweight, infrastructure-free visual SLAM algorithm. CooperSLAM leverages peer-to-peer communication and sparse map feature sharing to enable efficient collaborative localization and mapping in dynamic settings without requiring centralized resources or stable connectivity. A key innovation of CooperSLAM is the decoupling of map points and key frames on individual robots, combined with an opportunistic encountering mechanism to re-localize users and align maps. This design significantly enhances the scalability of multi-user AR, enabling its use in diverse scenarios.

1.3 Organization

In the rest of this manuscript, we introduce a fully distributed multi-robot pose estimation algorithm that utilizes both common point and common line features in Chapter 2.

Each robot not only leverages its own point and line measurements while also cooperating with neighboring robots to enhance localization accuracy. The effectiveness of the proposed approach is demonstrated through extensive simulations and experiments. In Chapter 3, we present a novel pose estimation model designed to initialize the first IMU state of VINS within the prior map frame. Additionally, we propose an online NeRF-based VIO algorithm that integrates a NeRF-based prior map with the online measurements from the camera. The performance of this method is validated using a real-world dataset. In Chapter 4, we propose an intelligent feature-based map alignment algorithm that facilitates sparse feature exchanges over intermittent peer-to-peer connections. The approach decouples map points from key frames and incorporates a lightweight data transmission mechanism tailored for multi-robot groups. Finally, Chapter 5 concludes the manuscript by summarizing the main findings and contributions.

Chapter 2

Point-Line Cooperative Visual-Inertial Odometry

2.1 Introduction and Related Works

SLAM has garnered significant attention over the past few decades, becoming a core technology in robotics and computer vision applications such as AR/VR [28], autonomous driving [100], and robot navigation [104]. In GPS-denied environments, VINS and related algorithms [74, 55, 93] have gained widespread popularity due to their ability to leverage low-cost and lightweight onboard cameras and IMUs.

In human-made scenarios, lines serve as valuable complements to points, especially in low-feature environments where only a limited number of point features can be extracted. There are two main categories of methods for processing points and lines in VINS: *indirect* (feature-based) and *direct* (epipolar constraint-based) methods. In particular, indirect

methods involve preprocessing image sequences by extracting feature descriptors and matching them across frames [74, 53, 43, 65, 70, 111]. These methods optimize the system by minimizing geometric errors, and ensuring accurate alignment of features over time. Direct methods, in contrast, bypass feature extraction and instead optimize photometric error directly using raw pixel intensities [34, 33, 107]. These methods are computationally efficient and benefit from leveraging epipolar constraints to maintain geometric consistency. However, direct methods typically assume brightness constancy (ignoring exposure changes), which makes them sensitive to real-world variations in exposure and lighting.

Among the previous feature-based VINS literature, the solutions can be broadly classified into two categories: filter-based methods [65, 21, 36, 71, 99, 96, 97, 103] and graph-based methods [74, 26, 43, 53, 73, 67]. One of the *state-of-the-art* works of the filter-based methods is MSCKF [65], which formed a multi-constraint update by using the measurements of the same feature. A key contribution of MSCKF is the left-nullspace projection, which efficiently marginalizes feature measurements from the state vector, retaining only the robot's state within the current sliding window. A tightly coupled monocular graph-based VIO (VINS-Mono) and nonlinear optimization with robust initialization introduced in [74]. Besides, there are also some VINS algorithms using both point and line features. The point-line visual-inertial odometry (PL-VIO) [43] is an extension of VINS-Mono, which can optimize the re-projection errors of the point and line features in a sliding window. PL-SLAM [73] proposed a point-line SLAM framework based on ORB-SLAM [67]. Line features used in Plücker representation for rolling-shutter cameras were designed in [99]. Furthermore, article [96] proposed two line triangulation algorithms. An in-depth analysis

of three line representations (Plücker, Quaternion, Closest Point) along with their respective observability properties was provided in [97].

One notable advantage of cooperative VINS (C-VINS) is the ability to share common features across multiple robots, thereby introducing additional geometric constraints on these shared features. Unlike traditional approaches where each robot only processes its own measurements, C-VINS enables robots to also incorporate observations from the entire multi-robot group. By leveraging these shared constraints, robots can enhance localization performance through updates informed by the collective data. However, a key question for a multi-robot group is how to best utilize the environment information and other robots' information.

Several centralized multi-robot solutions, such as those proposed in [61, 57, 47, 94], have been developed. However, these approaches often demand significant computational resources and extensive communication overhead. In contrast, distributed algorithms offer distinct advantages in terms of scalability and efficiency. For instance, [20] presented a distributed point-line cooperative SLAM (C-SLAM) algorithm using the M-Space representation for various features. However, this approach struggled with estimation consistency due to repeated utilization of the same information within the robot group. Similarly, in [48], they proposed a method where each robot processed its own measurements and fused estimations and covariances with other robots within communication range, but only at specific time steps. More recently, DOOR-SLAM [52] introduced a fully distributed C-SLAM algorithm featuring a pose graph optimizer and a data-efficient SLAM frontend, akin to the approach in [31]. Additionally, [69] proposed a fully distributed algorithm based on the

maximum a posteriori (MAP) estimation. CVIO [108] contributed a fully distributed cooperative algorithm that ensures consistency by employing the covariance intersection (CI) update. However, CVIO did not address challenges posed by low-feature environments, leaving room for further refinement in such scenarios.

These observations inspire the development of a fully distributed algorithm that leverages both point and line features in the environment. In our approach (PL-CVIO), each robot not only utilizes its own point and line measurements but also collaborates with its neighbors to enhance localization accuracy as shown in Figure 2.1. This cooperation is particularly beneficial in low-feature environments, where robust landmarks are sparse. By fusing independent point and line features from each robot and applying the CI update, our method effectively exploits the constraints introduced by commonly observed features shared among neighbors. Built upon the foundation of the original MSCKF algorithm, we extend it by incorporating closest point line constraints to maximize its robustness and precision.

2.2 Preliminaries

2.2.1 JPL Quaternion

The JPL quaternion [101] is defined as the following linear combination:

$$\bar{q} = q_4 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}, \tag{2.1}$$

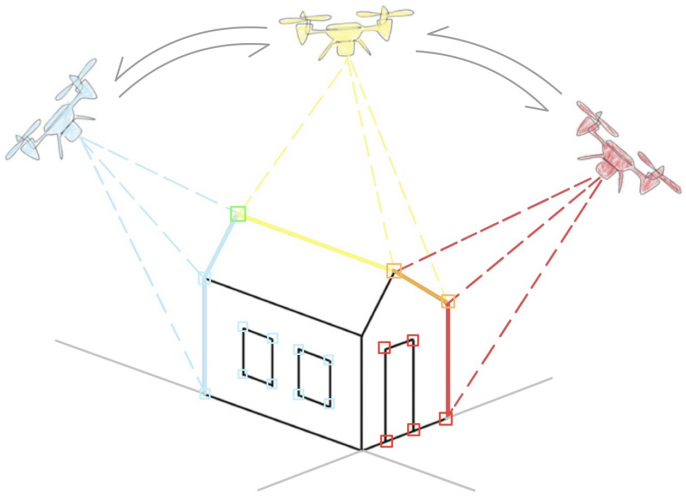


Figure 2.1: Overview of the PL-CVIO. Multiple robots observe point (square) and line (line segment) features in the same environment, neighbors communicate and share common points (green and orange squares) and common lines (orange line).

where $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ satisfy the quaternion algebra:

$$\begin{gathered} \mathbf{i}^2 = -1, \quad \mathbf{j}^2 = -1, \quad \mathbf{k}^2 = -1 \\ -\mathbf{ij} = \mathbf{ji} = \mathbf{k}, \quad -\mathbf{jk} = \mathbf{kj} = \mathbf{i}, \quad -\mathbf{ki} = \mathbf{ik} = \mathbf{j}. \end{gathered} \tag{2.2}$$

Here, q_4 represents the real (or scalar) component of the quaternion, while q_1, q_2, q_3 make up the imaginary (or vector) components. To simplify the notation, we split the quaternion as follows:

$$\bar{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^\top = \begin{bmatrix} \mathbf{q} \\ q_4 \end{bmatrix}. \tag{2.3}$$

If rotate an angle θ around the axis $\mathbf{k}$, the rotation can be represented by a unit quaternion as:

$$\mathbf{q} = \begin{bmatrix} k_x \sin(\theta/2) \\ k_y \sin(\theta/2) \\ k_z \sin(\theta/2) \end{bmatrix} = \hat{\mathbf{k}} \sin(\theta/2), \quad q_4 = \cos(\theta/2) \tag{2.4}$$

where the elements $q_1, \ldots, q_4$ are called *quaternion of rotation* or *Euler symmetric parameters* [88]. A quaternion of rotation is a unit quaternion, satisfying:

$$|\bar{q}| = \sqrt{\bar{q}^\top \bar{q}} = \sqrt{|\mathbf{q}|^2 + q_4^2} = 1. \tag{2.5}$$

The relationship between a unit quaternion and its corresponding rotational matrix is given by:

$$\mathbf{R}(\bar{q}) = \left(2q_4^2 - 1\right) \mathbf{I}_{3\times 3} - 2q_4[\mathbf{q}\times] + 2\mathbf{q}\mathbf{q}^\top, \tag{2.6}$$

where $\mathbf{I}_{3\times 3}$ denotes the 3×3 identity matrix, and $\lfloor \cdot \times \rfloor$ denotes the skew-symmetric matrix as:

$$\lfloor \boldsymbol{q} \times \rfloor = \begin{bmatrix} 0 & -q_z & q_y \\ q_z & 0 & -q_x \\ -q_y & q_x & 0 \end{bmatrix}. \tag{2.7}$$

Besides, note that $\bar{q}$ and $-\bar{q}$ describe the same rotation.

2.2.2 Notations and Definitions

Let the vector $\mathbf{x}$ represent the true state of a robot. The error state is defined as $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$, where $\hat{(\cdot)}$ denotes the estimated state and $\tilde{(\cdot)}$ denotes the error state. Then,

the $\mathbf{x}_{k|k-1}$ and $\mathbf{x}_{k|k}$ denote the prior and posterior estimates of the state at timestamp k, respectively. And $\mathbf{P}_{k|k-1}$ and $\mathbf{P}_{k|k}$ represent the covariance matrix corresponding to the prior and posterior state estimates at timestamp k, respectively.

We define the global frame $\{G\}$, the IMU frame $\{I\}$, and the camera frame $\{C\}$. To facilitate the cooperative case, $\{I_i\}$ and $\{C_i\}$ are the IMU and camera frames of robot i, for $i = 1, \cdots, n$, respectively. Additionally, $\mathbf{x}_{i,k}$ represents the state of robot i at timestamp k, and ${}^{G}\mathbf{x}_{i,k}$ denotes the state of robot i at timestamp k in the global frame.

2.3 Problem Formulation

The objective of the point and line cooperative visual-inertial odometry (PL_CVIO) is to estimate and track the 3D pose of each robot in the global frame by leveraging both point and line features. Unlike independent visual-inertial odometry, PL_CVIO allows multiple robots to share common features (common point and common line) with their neighbors, which collaboratively enhances localization accuracy.

2.3.1 Visual-Inertial Odometry State Vector

In order to perform the PL-CVIO, the state vector of each robot i is defined as:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{x}_{I_i}^\top & \mathbf{x}_{Calib_i}^\top & \mathbf{x}_{C_i}^\top & t_{d_i} \end{bmatrix}^\top, \tag{2.8}$$

where $\mathbf{x}_{I_i}$ denotes the IMU state vector, $\mathbf{x}_{Calib_i}$ denotes the rigid body tranformation between the IMU frame and camera frame, $\mathbf{x}_{C_i}$ represents the cloned IMU states, and $t_{d_i} = t_{C_i} - t_{I_i}$ denotes the time-offset between robot i's camera clock and IMU clock, which

treats the IMU clock as the true time [56, 75]. At any time step k, the state vector of each IMU can be writen as:

$$\mathbf{x}_{I_{i,k}} = \begin{bmatrix} {}^{I_{i,k}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{i,k}}^\top & {}^{G}\mathbf{v}_{I_{i,k}}^\top & \mathbf{b}_{g_{i,k}}^\top & \mathbf{b}_{a_{i,k}}^\top \end{bmatrix}^\top, \tag{2.9}$$

where ${}^{I_{i,k}}_{G}\bar{q}$ denotes the JPL unit quaternion representing the rotation from the global frame to the IMU frame at time step k. ${}^{G}\mathbf{p}_{I_{i,k}}$ and ${}^{G}\mathbf{v}_{I_{i,k}}$ are the IMU position and velocity in the global frame at time step k. $\mathbf{b}_{g_{i,k}}$ and $\mathbf{b}_{a_{i,k}}$ are the gyroscope and accelerometer biases at time step k. Then, the error state of the IMU is defined as:

$$\tilde{\mathbf{x}}_{I_{i,k}} = \begin{bmatrix} \delta^{I_{i,k}}_{G}\boldsymbol{\theta}^\top & {}^{G}\tilde{\mathbf{p}}_{I_{i,k}}^\top & {}^{G}\tilde{\mathbf{v}}_{I_{i,k}}^\top & \tilde{\mathbf{b}}_{g_{i,k}}^\top & \tilde{\mathbf{b}}_{a_{i,k}}^\top \end{bmatrix}^\top, \tag{2.10}$$

where the position, velocity, and bias errors utilize the standard additive error, while the quaternion error state is described by

$$\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}} \simeq \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}^\top & 1 \end{bmatrix}^\top \otimes \hat{\bar{q}}, \tag{2.11}$$

where $\otimes$ is the quaternion multiplication operator.

In addition to robot i's IMU state, the spatial calibration between its IMU frame and camera frame will also be estimated. In particular, the calibration state vector contains the unit quaternion rotation from the IMU frame to the camera frame ${}^{C_i}_{I_i}\bar{q}$, and the

translation from the IMU frame to the camera frame ${}^{C_i}\mathbf{p}_{I_i}$ as:

$$\mathbf{x}_{Calib_i} = \begin{bmatrix} {}^{C_i}_{I_i}\bar{q}^\top & {}^{C_i}\mathbf{p}_{I_i}^\top \end{bmatrix}^\top. \tag{2.12}$$

Robot i maintains a sliding window with m cloned IMU poses at time step k written as:

$$\mathbf{x}_{C_{i,k}} = \begin{bmatrix} {}^{I_{i,k-1}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{i,k-1}}^\top & \cdots & {}^{I_{i,k-m}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{i,k-m}}^\top \end{bmatrix}^\top. \tag{2.13}$$

2.3.2 Dynamic System Model

For each robot i, the measurement of the IMU linear acceleration ${}^{I_i}\mathbf{a}_m$ and the angular velocity ${}^{I_i}\boldsymbol{\omega}_m$ are modeled as:

$${}^{I_i}\mathbf{a}_m = {}^{I_i}\mathbf{a} + {}^{I_i}_{G}\mathbf{R}\,{}^{G}\mathbf{g} + \mathbf{b}_{a_i} + \mathbf{n}_{a_i}, \tag{2.14}$$

$${}^{I_i}\boldsymbol{\omega}_m = {}^{I_i}\boldsymbol{\omega} + \mathbf{b}_{g_i} + \mathbf{n}_{g_i}, \tag{2.15}$$

where ${}^{I_i}\mathbf{a}$ and ${}^{I_i}\boldsymbol{\omega}$ are the true angular velocity and linear acceleration. $\mathbf{n}_{a_i}$ and $\mathbf{n}_{g_i}$ represent the continuous-time Gaussian noises that contaminate the IMU measurements. ${}^{G}\mathbf{g}$ denotes the gravity expressed in the global frame. Then, the dynamic system of each IMU can be modeled as [88]:

$$\begin{aligned} &{}^{I_i}_{G}\dot{\bar{q}}(t) = \frac{1}{2}\boldsymbol{\Omega}\left({}^{I_i}\boldsymbol{\omega}(t)\right){}^{I_i}_{G}\bar{q}(t), \quad \dot{\mathbf{b}}_{g_i}(t) = \mathbf{n}_{wg_i}(t), \\ &{}^{G}\dot{\mathbf{v}}_{I_i}(t) = {}^{G}\mathbf{a}_i(t), \quad \dot{\mathbf{b}}_{a_i}(t) = \mathbf{n}_{wa_i}(t), \quad {}^{G}\dot{\mathbf{p}}_{I_i}(t) = {}^{G}\mathbf{v}_{I_i}(t) \end{aligned} \tag{2.16}$$

where ${}^{G}\mathbf{a}_i$ is the body acceleration in the global frame. ${}^{G}\mathbf{v}_{I_i}$, ${}^{G}\mathbf{p}_{I_i}$ are the velocity and position of the IMU in the global frame. $\mathbf{n}_{wg_i}$ and $\mathbf{n}_{wa_i}$ denote the zero-mean Gaussian noises driving the IMU biases. $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^\top$ is the rotational velocity in the IMU frame and

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor \boldsymbol{\omega} \times \rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}.$$

After linearization, the continuous-time IMU error-state can be written as:

$$\dot{\tilde{\mathbf{x}}}_i(t) \simeq \mathbf{F}_i(t)\tilde{\mathbf{x}}_i(t) + \mathbf{G}_i(t)\mathbf{n}_i(t), \tag{2.17}$$

where $\mathbf{F}_i(t)$ is the 15×15 continuous-time IMU error-state Jacobian matrix, $\mathbf{G}_i(t)$ is the 15×12 noise Jacobian matrix, and $\mathbf{n}_i(t) = \left[\mathbf{n}_{g_i}^\top\ \mathbf{n}_{wg_i}^\top\ \mathbf{n}_{a_i}^\top\ \mathbf{n}_{wa_i}^\top\right]^\top$ is the system noise with the covariance matrix $\mathbf{Q}_i$.

In order to propagate the covariance matrix from discrete-time t_k to t_{k+1}, the state transition matrix $\boldsymbol{\Phi}_i\left(t_{k+1}, t_k\right)$ is computed by solving the differential equation:

$$\dot{\boldsymbol{\Phi}}_i\left(t_{k+1}, t_k\right) = \mathbf{F}_i \boldsymbol{\Phi}_i\left(t_{k+1}, t_k\right), \tag{2.18}$$

with the initial condition $\boldsymbol{\Phi}_i\left(t_k, t_k\right) = \mathbf{I}_{15}$. Thus, the discrete-time noise covariance can be expressed as:

$$\mathbf{Q}_{i,k} = \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}_i(t_{k+1}, \tau)\mathbf{G}_i(\tau)\mathbf{Q}_i\mathbf{G}_i^\top(\tau)\boldsymbol{\Phi}_i(t_{k+1}, \tau)^\top \mathbf{d}\tau, \tag{2.19}$$

and the propagated covariance can be written as:

$$\mathbf{P}_{i,k+1|k} = \mathbf{\Phi}_i\left(t_{k+1}, t_k\right)\mathbf{P}_{i,k|k}\mathbf{\Phi}_i\left(t_{k+1}, t_k\right)^\top + \mathbf{Q}_{i,k}. \tag{2.20}$$

2.3.3 Point and Line Measurement Models

In low-feature environments, lines are good complements to points. Hence we consider both point and line measurements in this paper. The point measurements of robot i can be described by:

$${}^{C_i}\mathbf{z}_p = \Pi\left({}^{C_i}\mathbf{x}_p\right) + \mathbf{w}_{p_i}, \quad \Pi\left(\left[x\ y\ z\right]^\top\right) = \begin{bmatrix}\frac{x}{z} & \frac{y}{z}\end{bmatrix}^\top, \tag{2.21}$$

where ${}^{C_i}\mathbf{x}_p$ is the 3D position of the point in the camera frame, and $\mathbf{w}_{p_i}$ denotes the corresponding measurement noise. Based on the relative transformation and time offset definition in (3.3), the relationship between point feature in the global frame ${}^{G}\mathbf{x}_p$ and in the camera frame ${}^{C_i}\mathbf{x}_p$ can be expressed as:

$${}^{C_i}\mathbf{x}_p = {}^{C_i}_{I_i}\mathbf{R}\,{}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right)\left({}^{G}\mathbf{x}_p - {}^{G}\mathbf{p}_{I_i}\left(\bar{t}_i\right)\right) + {}^{C_i}\mathbf{p}_{I_i}, \tag{2.22}$$

where $\bar{t}_i = t_i - t_{d_i}$ is the exact camera time of the relative transformation between the global frame and the IMU frame.

For a 3D line, we adopt the *Closest Point* representation [96], which represents the 3D line by multiplying a unit quaternion and the corresponding distance scalar from the origin to this line. Given the 3D positions of two points $\mathbf{p_{f1}}$ and $\mathbf{p_{f2}}$ on a line, the Plücker

coordinate can be expressed by [112]:

$$\begin{bmatrix} \mathbf{n}_l \\ \mathbf{v}_l \end{bmatrix} = \begin{bmatrix} \lfloor \mathbf{p}_{\mathbf{f}1} \times \rfloor \, \mathbf{p}_{\mathbf{f}2} \\ \mathbf{p}_{\mathbf{f}2} - \mathbf{p}_{\mathbf{f}1} \end{bmatrix}, \tag{2.23}$$

where $\mathbf{n}_l$ denotes the normal direction of the line-plane and $\mathbf{v}_l$ is the line direction. Then, the *Closest Point* line can be expressed as:

$${}^{G}\mathbf{x}_l = d_l \bar{q}_l = \begin{bmatrix} \mathbf{q}_l^\top & q_l \end{bmatrix}^\top, \tag{2.24}$$

where the distance scalar can be computed as $d_l = \|\mathbf{n}_l\| / \|\mathbf{v}_l\|$. The unit quaternion $\bar{q}_l$ can be transformed from $\mathbf{R}(\bar{q}_l) = [\mathbf{n_e} \ \mathbf{v_e} \ \lfloor \mathbf{n_e} \times \rfloor \mathbf{v_e}]$, where $\mathbf{n_e}$ and $\mathbf{v_e}$ are the unit 3D vectors of $\mathbf{n}_l$ and $\mathbf{v}_l$.

Moreover, for each robot i, we adopt the simple projective line measurement model [18] to describe the 2D line distance from two line endpoints, $\mathbf{x}_{s_i} = [u_{s_i} \ v_{s_i} \ 1]^\top$ and $\mathbf{x}_{e_i} = [u_{e_i} \ v_{e_i} \ 1]^\top$ to the 2D line segment:

$${}^{C_i}\mathbf{z}_l = \begin{bmatrix} \frac{\mathbf{x}_{s_i}^\top \mathbf{l}_i}{\sqrt{l_1^2 + l_2^2}} & \frac{\mathbf{x}_{e_i}^\top \mathbf{l}_i}{\sqrt{l_1^2 + l_2^2}} \end{bmatrix}^\top, \tag{2.25}$$

where $\mathbf{l}_i = [l_1 \ l_2 \ l_3]^\top$ denotes the 2D line representation. The line measurement can be

projected from the 3D line in the camera frame as in [97]:

$$\begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \begin{bmatrix} f_{v_i} & 0 & 0 & 0 & 0 & 0 \\ 0 & f_{u_i} & 0 & 0 & 0 & 0 \\ -f_{v_i}c_{u_i} & -f_{u_i}c_{v_i} & f_{u_i}f_{v_i} & 0 & 0 & 0 \end{bmatrix} {}^{C_i}\mathbf{L}, \tag{2.26}$$

where f_{u_i}, f_{v_i}, c_{u_i}, c_{v_i} are the camera intrinsic parameters, and ${}^{C_i}\mathbf{L} = \begin{bmatrix} {}^{C_i}d_l {}^{C_i}\mathbf{n}_e^\top & {}^{C_i}\mathbf{v}_e^\top \end{bmatrix}^\top$ is the Plücker coordinate representation of the 3D line in the camera frame. The line transformation from the global frame to the camera frame can be written as:

$${}^{C_i}\mathbf{L} = \begin{bmatrix} {}^{C_i}_{I_i}\mathbf{R} & \lfloor {}^{C_i}\mathbf{P}_{I_i} \times \rfloor {}^{C_i}_{I_i}\mathbf{R} \\ \mathbf{0}_3 & {}^{C_i}_{I_i}\mathbf{R} \end{bmatrix} {}^{I_i}\mathbf{L}$$

and

$${}^{I_i}\mathbf{L} = \begin{bmatrix} {}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right) & -{}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right) \lfloor {}^{G}\mathbf{P}_{I_i}\left(\bar{t}_i\right) \times \rfloor \\ \mathbf{0}_3 & {}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right) \end{bmatrix} {}^{G}\mathbf{L}, \tag{2.27}$$

where ${}^{I_i}\mathbf{L}$ and ${}^{G}\mathbf{L}$ are the Plücker line representations in the IMU frame and the global frame, respectively.

2.3.4 Independent Point and Line Feature Update

To perform the independent point or line feature update, a standard MSCKF update [65] will be applied to each robot. In particular, we collect all of the point and line measurements over the current sliding window. By stacking the measurements of one point

or line, we can triangulate the point feature or line feature utilizing the estimate of the IMU poses. To simplify the notation, let $\tilde{\mathbf{x}}_f$ denotes either a point feature or a line feature, and the measurement residual of robot i can be linearized as:

$$\mathbf{r}_i = \mathbf{h}\left(\tilde{\mathbf{x}}_i, {}^G\tilde{\mathbf{x}}_f\right) + \mathbf{w}_i \simeq \mathbf{H}_{i,x}\tilde{\mathbf{x}}_i + \mathbf{H}_{i,f}{}^G\tilde{\mathbf{x}}_f + \mathbf{w}_i, \tag{2.28}$$

where $\mathbf{r}_i$ is the residual of a point or line measurement. $\mathbf{H}_{i,x}$ and $\mathbf{H}_{i,f}$ denote the Jacobians w.r.t. the state vector and the feature, respectively. $\mathbf{w}_i$ denotes the noise vector corresponding to the point or line feature.

After that, we perform the left nullspace projection by applying the QR decompositon to $\mathbf{H}_{i,f}$ in (2.28) as:

$$\begin{bmatrix} \mathbf{r}_i^1 \\ \mathbf{r}_i^2 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{i,x}^1 \\ \mathbf{H}_{i,x}^2 \end{bmatrix} \tilde{\mathbf{x}}_i + \begin{bmatrix} \mathbf{H}_{i,f}^1 \\ \mathbf{0} \end{bmatrix} {}^G\tilde{\mathbf{x}}_f + \begin{bmatrix} \mathbf{w}_i^1 \\ \mathbf{w}_i^2 \end{bmatrix}. \tag{2.29}$$

In this expression, $\mathbf{r}_i^2$ is only related to the state vector $\tilde{\mathbf{x}}_i$. Hence robot i will perform an EKF update using $\mathbf{r}_i^2$, while $\mathbf{r}_i^1$ will be dropped.

2.3.5 Common Point and Line Feature Update

Note that neighboring robots might observe a common point or line feature. Hence, we will further exploit both point and line feature constraints among neighbors to improve the localization accuracy. The robots can communicate with their neighbors to share information.

Robot i and its neighbors will apply the linearization (2.28) and the left nullspace projection (2.29) to the common feature, denoted as ${}^{G}\tilde{\mathbf{x}}_f$. As in Sec 2.3.4, robot i will use $\mathbf{r}_i^2$ for an EKF update. However, instead of dropping $\mathbf{r}_i^1$, robot i will exploit shared information from its neighbors. It will construct a new residual system that depends on the common point or line feature ${}^{G}\tilde{\mathbf{x}}_f$ by stacking the top parts in (2.29) associated with itself and its neighbors as in [108]:

$$\begin{bmatrix} \mathbf{r}_i^1 \\ \mathbf{r}_{i_1}^1 \\ \vdots \\ \mathbf{r}_{i_j}^1 \end{bmatrix} = diag\left(\begin{bmatrix} \mathbf{H}_{i,x}^1 \\ \mathbf{H}_{i_1,x}^1 \\ \vdots \\ \mathbf{H}_{i_j,x}^1 \end{bmatrix} \right) \begin{bmatrix} \tilde{\mathbf{x}}_i \\ \tilde{\mathbf{x}}_{i_1} \\ \vdots \\ \tilde{\mathbf{x}}_{i_j} \end{bmatrix} + \begin{bmatrix} \mathbf{H}_{i,f}^1 \\ \mathbf{H}_{i_1,f}^1 \\ \vdots \\ \mathbf{H}_{i_j,f}^1 \end{bmatrix} {}^{G}\tilde{\mathbf{x}}_f + \begin{bmatrix} \mathbf{w}_i^1 \\ \mathbf{w}_{i_1}^1 \\ \vdots \\ \mathbf{w}_{i_j}^1 \end{bmatrix}, \tag{2.30}$$

where *diag* denotes the block-diagonal matrix, and $i_1 \dots i_j$ denote the neighbors of robot i. Then, we utilize the left nullspace projection to the stacked common point or line feature Jacobian matrix in (2.30) and obtain a new residual system that is independent of the common feature as:

$$\mathbf{r}_i' = \begin{bmatrix} \mathbf{H}_{i,x}' & \mathbf{H}_{i_1,x}' & \cdots & \mathbf{H}_{i_j,x}' \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_i \\ \tilde{\mathbf{x}}_{i_1} \\ \vdots \\ \tilde{\mathbf{x}}_{i_j} \end{bmatrix} + \mathbf{w}_i'. \tag{2.31}$$

In order to guarantee the consistency of estimation, we adopt the CI-EKF algorithm in [108], where the weights of the CI are $\omega_i > 0,\ \omega_{i_l} > 0$, and $\omega_i + \sum_{l=1}^{j} \omega_{i_l} = 1$. The

Kalman gain of robot i is given by:

$$\mathbf{K}_i = \frac{\mathbf{P}_{i,k+1|k}\mathbf{H}_{i,x}'^{\top}}{\omega_i}\left(\sum_{r\in\mathcal{N}_i}\frac{1}{\omega_r}\mathbf{H}_{r,x}'\mathbf{P}_{r,k+1|k}\mathbf{H}_{r,x}'^{\top}+\mathbf{R}_i\right)^{-1}, \tag{2.32}$$

where $\mathcal{N}_i$ denotes the set of robot i's neighboring robots that the current common feature can be tracked, and $\mathbf{R}_i$ denote the covariance matrix associated with $\mathbf{w}_i'$. Then, the state correction of robot i can be written as:

$$\Delta\mathbf{x}_{i,k} = \mathbf{K}_i\mathbf{r}_i'. \tag{2.33}$$

The state covariance matrix of robot i is updated using the CI as:

$$\mathbf{P}_{i,k+1|k+1} = \frac{1}{\omega_i}\left(\mathbf{I}-\mathbf{K}_i\mathbf{H}_{i,x}'\right)\mathbf{P}_{i,k+1|k}. \tag{2.34}$$

2.4 Simulations and Experiments

In this section, we utilize Monte-Carlo simulations and real-world datasets to verify that (1) common line features can improve localization accuracy in cooperative cases, (2) line features can also improve the accuracy in independent cases. We compare our PL-CVIO algorithm with the previous works in Table 2.1 under two different environments, where *low-feature* scenes contain a few features and *rich-feature* scenes contain enough features. As shown in Table 2.1, P-VIO denotes the independent MSCKF algorithm [65], PL-VIO denotes the independent point-line MSCKF algorithm [96], P-CVIO denotes our previous work CVIO [108], IPL-CP-CVIO denotes the algorithm which not only utilizes independent

Table 2.1: Descriptions of various algorithms to be compared in the simulations and experiments, focusing on the usage of independent features and common features.

Algorithm	**Independent Features**	**Common Features**
P-VIO [65]	Points	✗
PL-VIO [96]	Points w/ Lines	✗
P-CVIO [108]	Points	Points
IPL-CP-CVIO	Points w/ Lines	Points
PL-CVIO [Ours]	Points w/ Lines	Points w/ Lines

point-line features from each robot but also collects the common point features from the neighbors, and PL-CVIO uses both independent and common point-line features.

2.4.1 Monte-Carlo Simulations

For our Monte-Carlo simulations, we utilize a group of three robots. *Robot 0* in the group follows the real trajectory of a dataset, and the trajectories of *robot 1* and *robot 2* are created by adding position and orientation offsets to the real one. After that, the 3D features and the corresponding 2D measurements are generated if the number of the point or line measurements is below the threshold in the current frame. Then, the constraints of the same feature from one robot and the constraints of the common features from neighbors are collected and utilized to update the current state.

The low-feature and rich-feature environments are divided by extracting different numbers of point features. In the rich-feature environments, the number of point features is 150 and the number of line features is 50 in each frame. We reduce the number of point features to 50 for the low-feature cases. For both of these two environments, we

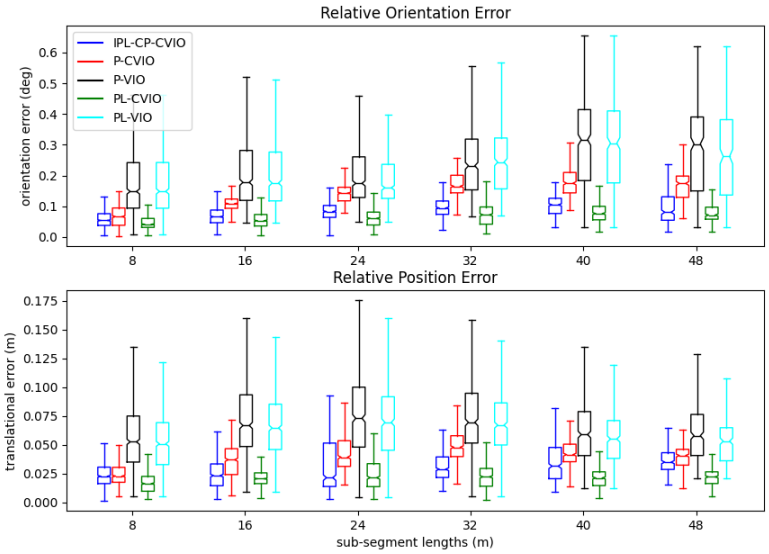


Figure 2.2: Boxplot of the statistics of the Monte-Carlo simulation under the rich-feature Udel_gore environment by extracing 150 points per frame, and 50 lines if the line update is used.

utilize the First-Estimation Jacobian (FEJ) and online camera-IMU calibration [39]. After running 30 Monte-Carlo loops, the statistics of the relative orientation error (ROE) and the relative position error (RPE) under the rich-feature or low-feature Udel_gore dataset are shown in Figure 2.2 and Figure 2.3, respectively. We can see that our PL-CVIO algorithm outperforms all other algorithms in both environments. Especially in the low-feature case, we can find out that the common line can reduce the ROE and RPE obviously (blue and red bar) as in Figure 2.3. Moreover, an interesting discovery is that PL-VIO outperforms P-CVIO if a limited number of points are observed in each frame. In this case, the number of common point features is also limited, and hence the cooperative method P-CVIO that relies on only common point features has limited resources to resort to. In contrast, the methods PL-VIO and PL-CVIO that further exploit line features exhibit better performance

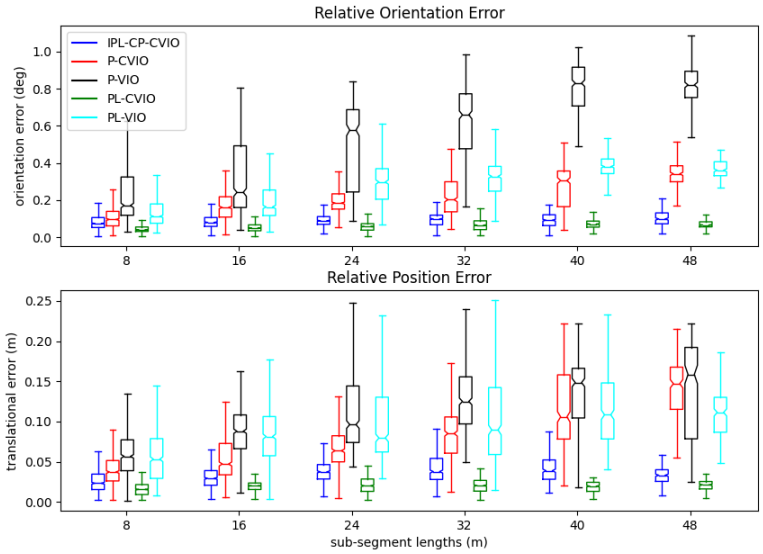


Figure 2.3: Boxplot of the statistics of the Monte-Carlo simulation under low-feature Udel_gore environment by extracing 50 points per frame, and 50 lines if the line update is used.

while PL-CVIO achieves the best performance as it exploits not only point and line features but also cooperation with neighbors.

Additionally, we simulate our PL-CVIO algorithm in all of the EuRoC V1 datasets [24] and compare it with P-VIO and P-CVIO in both low-feature and rich-feature environments. The RMSE of the orientation and position of each robot and the mean RMSE of each algorithm in each environment are recorded in Table 2.2 and Table 2.3. The RMSE results show that our PL-CVIO algorithm outperforms P-CVIO and P-VIO in all simulated scenarios. Especially in low-feature environments, the PL-CVIO improves the RMSE of orientation and position dramatically.

Table 2.2: The RMSE of the orientation / position (degrees / meters) of three robots using three different algorithms in **rich-feature** environments in different EuRoC datasets. The average denotes mean of all three rooms per algorithm per robot per environment. R0, R1, and R2 represent three robots following three different trajectories in each environment.

	V1_01	**V1_02**	**V1_03**	**Average**
R0 P-VIO	0.481/0.260	0.621/0.064	0.874/0.061	0.659/0.128
R0 P-CVIO	0.091/0.056	0.157/0.022	0.118/0.027	0.122/0.035
R0 PL-CVIO	0.090/0.047	0.147/0.021	0.101/0.025	0.113/0.031
R1 P-VIO	1.166/0.205	0.167/0.049	0.419/0.049	0.584/0.101
R1 P-CVIO	0.104/0.060	0.183/0.026	0.127/0.026	0.138/0.037
R1 PL-CVIO	0.089/0.052	0.176/0.021	0.096/0.026	0.120/0.033
R2 P-VIO	0.960/ 0.132	0.230/0.078	0.325/0.062	0.505/0.091
R2 P-CVIO	0.099/0.056	0.170/0.023	0.123/0.025	0.131/0.035
R2 PL-CVIO	0.095/0.056	0.167/0.022	0.109/0.021	0.127/0.033

Table 2.3: The RMSE of the orientation / position (degrees / meters) of three robots using three different algorithms in **low-feature** environments in different EuRoC datasets. The average denotes mean of all three rooms per algorithm per robot per environment. R0, R1, and R2 represent three robots following three different trajectories in each environment.

	V1_01	**V1_02**	**V1_03**	**Average**
R0 P-VIO	1.277/0.483	0.717/0.177	1.184/0.684	1.060/0.448
R0 P-CVIO	0.524/0.148	0.449/0.080	0.743/0.299	0.572/0.176
R0 PL-CVIO	0.159/0.078	0.167/0.064	0.182/0.099	0.169/0.080
R1 P-VIO	0.888/0.152	0.785/0.170	0.775/0.150	0.816/0.157
R1 P-CVIO	0.584/0.137	0.613/0.130	0.733/0.075	0.643/0.144
R1 PL-CVIO	0.213/0.092	0.231/0.078	0.249/0.074	0.231/0.081
R2 P-VIO	1.589/0.596	1.493/0.195	0.676/0.202	1.253/0.331
R2 P-CVIO	0.603/0.161	0.690/0.179	0.538/0.165	0.610/0.168
R2 PL-CVIO	0.150/0.096	0.167/0.081	0.182/0.073	0.166/0.083

2.4.2 Experiments

For the real-world experiments, the position and orientation of each robot are initialized corresponding to the ground truth. The point features are extracted from each frame using FAST [78], and are tracked crossing frames or matched with the point observations from other robot utilizing ORB [79] with an 8-point RANSAC algorithm [59]. At the same time, line segments are extracted by leveraging the LSD [42] and tracked by LBD [102], as shown in Figure 2.4. Additionally, we add some outlier elimination strategies to remove the line segment where (1) the LBD distance is larger than 50; (2) the length of the line segment is smaller than 50 pixels; (3) the distance between the origin and this line is smaller than 0.1 or larger than 100; (4) the line disparity is too small to avoid singularity when applying the SVD [23] to triangulate this line.

We evaluate our PL-CVIO algorithm using the TUM Visual-Inertial Dataset Rooms 1, 3, and 5 [87], where the IMU operates at 200 Hz and the camera at 20Hz. Each dataset for a given room is loaded separately, and all five algorithms are tested independently on three robots. Besides, we extract a different number of point features to imitate low-feature and rich-feature environments. As a result, we show the experimental results of our PL-CVIO algorithm compared with the other four algorithms in respectively rich-feature environments as in Figure 2.5 and low-feature environments as in Figure 2.6. We also show the RMSE of the orientation and position of each robot by utilizing different algorithms in the TUM dataset as in Table 2.4. From the ROE/RPE and the RMSE results, it is evident that line features can improve the accuracy of P-VIO and the common point-line features can improve the performance of the P-CVIO. Specifically, the improvement is particularly sig-

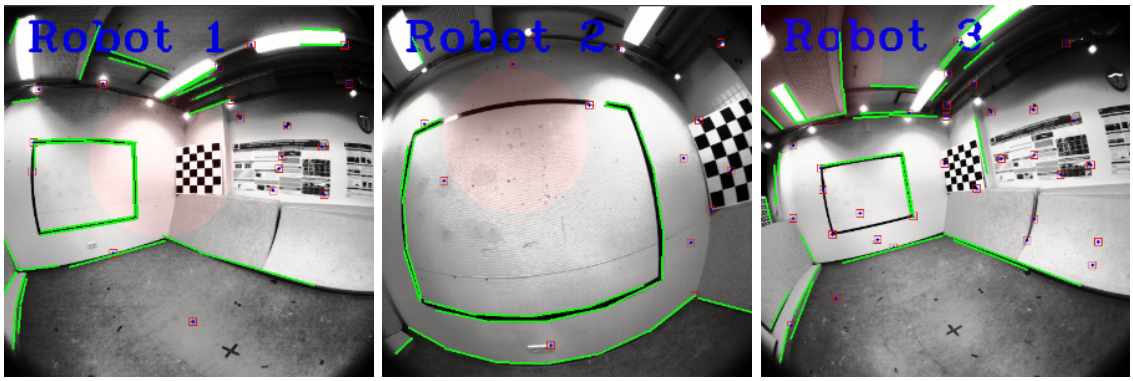


Figure 2.4: Point and line feature detection of three different robots in the TUM dataset [87]. A green edge denotes a line extracted from the current frame, and a blue dot surrounded by a red square denotes a point extracted from the current frame.

nificant in low-feature environments, as observed when comparing P-VIO with PL-VIO, and P-CVIO with IPL-CP-CVIO in Table 2.4. Overall, our PL-CVIO algorithm consistently outperforms all four competing algorithms across all experimental scenarios.

2.5 Conclusions

In this chapter, we have proposed a fully distributed point-line cooperative visual-inertial navigation system. We compared the performance of the proposed algorithm with four other algorithms under rich-feature or low-feature environments in both Monte-Carlo simulations and real-world datasets. All of the results indicated that our PL-CVIO outperformed the independent MSCKF and CVIO. Also, we verified that the line feature can improve the accuracy of localization in independent cases, and the common line features can perform better in cooperative cases.

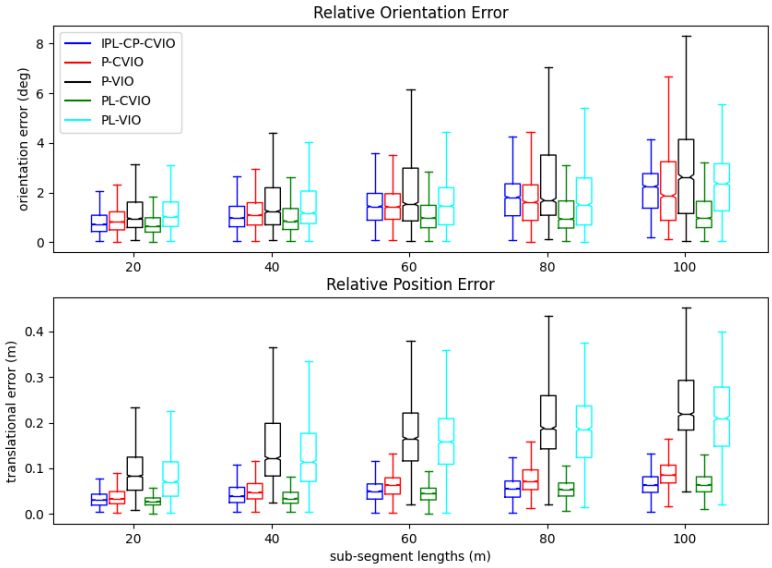


Figure 2.5: Boxplot of the result of *Robot 0* (Room 1) in the TUM Visual-Inertial Dataset by extracing 200 points per frame, and 50 lines if the line update is used.

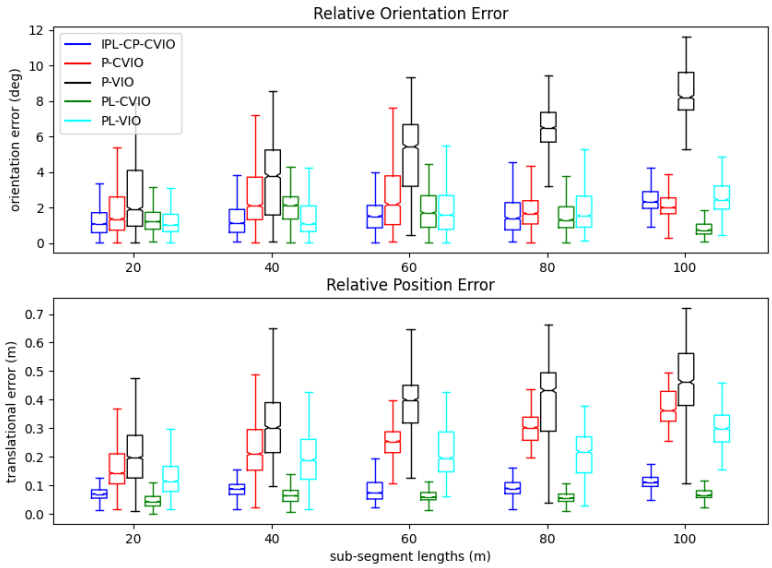


Figure 2.6: Boxplot of the result of *Robot 0* (Room 1) in the TUM Visual-Inertial Dataset by extracing 50 points per frame, and 50 lines if the line update is used.

Table 2.4: The RMSE of the orientation / position (degrees / meters) of three robots under the low-feature environments by using five different algorithms in the TUM Visual-Inertial dataset.

Algorithm	**Robot 0**	**Robot 1**	**Robot 2**
P-VIO	7.473 / 0.442	4.357 / 0.313	5.468 / 0.372
PL-VIO	2.367 / 0.295	1.798 / 0.254	1.872 / 0.240
P-CVIO	2.301 / 0.377	3.824 / 0.267	2.616 / 0.263
IPL-CP-CVIO	1.905 / 0.098	1.722 / 0.115	1.542 / 0.095
PL-CVIO	1.349 / 0.061	1.665 / 0.086	1.379 / 0.067

Chapter 3

Map-Based Visual-Inertial Odometry Leveraging Neural Radiance Fields

3.1 Introduction and Related Works

Recently, AR [2, 7] and VR [4, 1] have emerged as transformative technologies, offering immersive experiences across various domains. One critical aspect shaping the effectiveness of these experiences is the incorporation of prior maps [91]. These maps provide essential spatial context, enabling accurate localization, tracking, and seamless integration of virtual elements into the real world. To achieve high quality and low latency user experiments, VINS have received considerable popularity in AR/VR applications [65, 74, 39, 108, 105] through utilizing low-cost and lightweight onboard cameras and IMUs.

Using VINS, the drift of the pose will accumulate and the uncertainty of the estimate will grow unbounded without global information, such as a prior map, GNSS measurement, or loop closure. However, GNSS may not be applicable indoors, and loop closure demands both a precise and efficient place recognition algorithm [38, 84, 30] and substantial memory space to store historical features [67, 68]. Consequently, prior map-based approaches have gained significant interest over the past few decades [49, 82, 80, 40, 32, 106].

One of the key challenges that the map-based VINS literature tackles is relocalization based on one image and a prior map. Typically, descriptor-based methods are employed to establish 2D-3D correspondences by reprojecting map points to the image frame and matching them with features extracted from the image [58, 41]. Considering the increase in optimization complexity with map size, DBoW [38] represents an image by the statistic of different kinds of features from a visual vocabulary. Inspired by the DBoW, keyframe-based loop closure detection and localization are employed in ORB-SLAM [67] and ORB-SLAM2 [68]. However, DBoW sacrifices spatial information about features, potentially leading to ambiguities or inaccuracies.

Recently, NeRF [63] introduces an MLP to capture a radiance field representation of a scene. During training, NeRF estimates the color and density of sampled particles along each ray, and minimizes the photometric error between the estimated image and the groundtruth. NICE-SLAM [110] proposes a dense simultaneous localization and mapping (SLAM) system that incorporates depth information and minimizes depth loss during training. Subsequently, NICER-SLAM [109] further incorporates monocular normal estimators

and introduces a keyframe selection strategy. To expedite the training procedure, Nvidia Corp. proposes Instant-NGP [66], which utilizes a versatile new input encoding, enabling the use of a smaller network without compromising quality. Despite the notable enhancement in training speed, there is no assurance of compatibility with online VIO and NeRF map updates.

Among the NeRF-based localization literature, Loc-NeRF [62] introduces a real-time visual odometry (VO) algorithm by combining a particle filter with a NeRF prior map, which is trained offline. VO propagates the state of the pose, while rendered images from NeRF are used for updates. Due to the large number of particles and rendering costs from the NeRF model, Loc-NeRF operates at a much lower frequency of 0.6 Hz compared to the normal camera rate. NeRF-VINS [50] proposes a real-time VINS framework by integrating OpenVINS [39] and NeRF [63], utilizing both real and rendered images for updates at varying frequencies. *Nonetheless, none of the approaches above addresses pose initialization at the first timestamp. In other words, they assume the rigid transformation between the prior map frame and the online camera frame is known.* The only map-based relocalization work is iNeRF [98], they invert the NeRF pipeline and propose a gradient-based pose estimator by inputting a single image and a pre-trained NeRF model, but it heavily relies on a good initial guess.

To tackle the challenges outlined above, we proposes a real-time map-based VIO algorithm with pose initialization as in Figure 3.1. Specifically, we introduce an initialization model to estimate the first IMU state and a NeRF model to update the poses during traveling. For the initialization, we introduce an MLP-based model, which establishes the

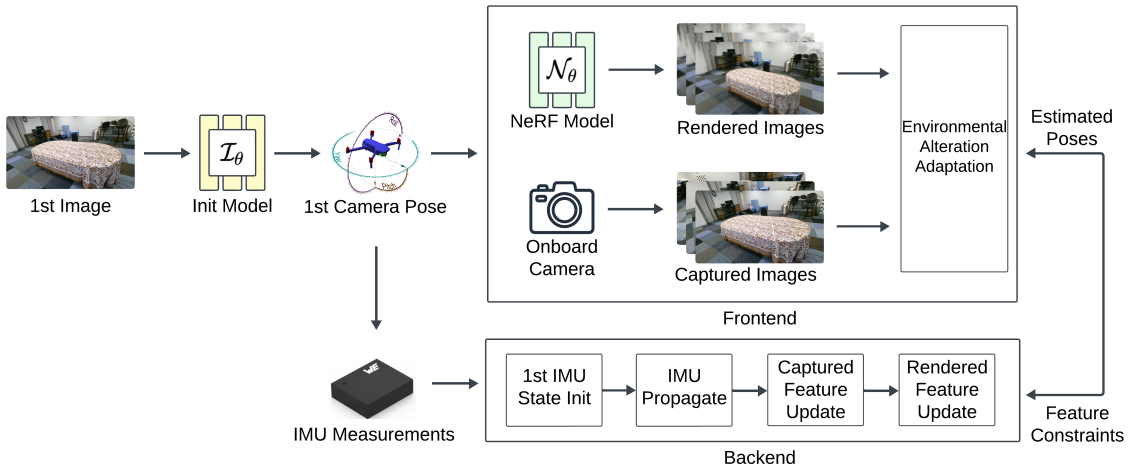


Figure 3.1: An overview of our NeRF-VIO framework. Commencing with the initial captured image, the pre-trained initialization model (canary) outputs the first pose of the camera frame. Utilizing IMU integration from the timestamp of the initial IMU measurement to that of the first camera measurement, we deduce the initial IMU state backward. Throughout online traveling, we leverage both the pre-trained NeRF model (mint) and the onboard camera to establish spatial constraints, facilitating the update of poses within the current sliding window. These updated poses then undergo further IMU propagation, serving as input to the NeRF model for the rendering of subsequent images.

correlations between images and poses without necessitating an initial guess. We define a novel loss function as the geodesic errors on $SE(3)$ and construct a left-invariant metric on $\mathfrak{se}(3)$. Additionally, we train a NeRF model capable of rendering images from new poses. During online traversal, the onboard camera captures images while the NeRF model renders images based on the estimated poses from VIO. These two pipelines operate concurrently but at different frequencies. Upon receiving a new rendered image, an object removal strategy is deployed to environmental alterations between the real world and the prior map. Subsequently, both captured and rendered images are utilized to update the robot's state.

3.2 Preliminaries

3.2.1 NeRF Map Generation and Image Rendering

Neural radiance fields (NeRF) [63] employs a multilayer perceptron (MLP) to capture a radiance field representation of a scene and generate images from new perspectives. The NeRF model can be trained offline given a sequence of RGB images and the corresponding 3D location and the 2D viewing direction, where the 2D viewing direction can be expressed as a 3D Cartesian unit vector. Once we get the NeRF map $\mathcal{N}_\theta$, a new image from a novel pose can be generated and each pixel on the image is predicted by projecting a ray $\boldsymbol{r}$ from the center of the camera to the position of this pixel on the image plane. Then some particles are sampled uniformly within $[t_n, t_f]$ along the ray and part of them are selected based on the estimated density σ. Finally, the color value of this pixel is rendered based on those selected particles as:

$$\hat{\mathcal{C}}(\boldsymbol{r}) = \int_{t_n}^{t_f} \hat{T}(\boldsymbol{r}, t)\hat{\sigma}(\boldsymbol{r}, t)\hat{\boldsymbol{c}}(\boldsymbol{r}, t)dt, \tag{3.1}$$

where $\hat{(\cdot)}$ denotes the estimated value and $\boldsymbol{c}$ denotes the RGB color to be predicted at one particle. The accumulated transmittance follows $\hat{T}(\boldsymbol{r}, t) = \exp\left(-\int_{t_n}^{t} \hat{\sigma}\left(\boldsymbol{r}, s\right) ds\right)$. Then, the loss function can be defined as:

$$\mathcal{L}(\mathcal{N}_\theta) = \sum_{\mathbf{r}\in\mathcal{R}} \|\hat{\mathcal{C}}(\mathbf{r}) - \mathcal{C}(\mathbf{r})\|_2^2, \tag{3.2}$$

where $\mathcal{R}$ denotes the set of rays. For a more comprehensive description, readers are referred to [63].

3.2.2 Notations and Definitions

Let the vector $\mathbf{x}$ represent the true state of a robot. The error state is defined as $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$, where $\hat{(\cdot)}$ denotes the estimated state and $\tilde{(\cdot)}$ denotes the error state. Then, the $\mathbf{x}_{k|k-1}$ and $\mathbf{x}_{k|k}$ denote the prior and posterior estimates of the state at timestamp k, respectively. And $\mathbf{P}_{k|k-1}$ and $\mathbf{P}_{k|k}$ represent the covariance matrix corresponding to the prior and posterior state estimates at timestamp k, respectively.

We define the global frame $\{G\}$, the prior map frame $\{W\}$, the IMU frame $\{I\}$, and the camera frame $\{C\}$. After relocalization is completed and the relative transformation between the global frame and the prior map frame is determined, they will share a common coordinate system.

3.3 Problem Formulation

The goal of the NeRF-VIO is to estimate the 3D pose of the IMU frame in the global frame given an initialization model $\mathcal{I}_\theta$ and a prior map $\mathcal{N}_\theta$. Specifically, the prior map is encoded by a NeRF model, which is trained offline using the image-pose pairs from a different trajectory in the same environment. As illustrated in Fig. 3.2, the initialization model is designed to relocalize a captured image from a prior map, while the NeRF model renders an image based on the current pose. The initialization model runs only once before online traversal. Note that the NeRF prior map resides within its own world frame, which

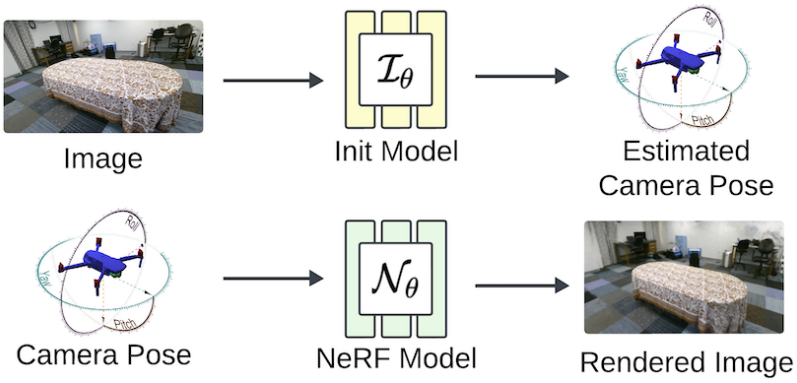


Figure 3.2: Comparison of input and output during model inference. The Init model estimates the camera pose in the world frame of a prior map based on a captured image. Conversely, the NeRF model renders an image when provided with a specific camera pose.

is not coincident with the global frame before initialization. During online traveling, the robot updates its state using both images rendered from the NeRF map and the captured images from the onboard cameras in the camera frame. We assume the sensor platform is pre-calibrated, ensuring that the relative transformation between the IMU frame and camera frame, denoted as $\mathbf{T}_I^C$, is already determined.

3.3.1 NeRF-VIO State Vector

To perform the NeRF-VIO, we include the IMU state, cloned IMU state, SLAM feature state, calibration state, and camera and IMU time-offset in the robot's state vector as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_{Clone}^\top & \mathbf{x}_f^\top & \mathbf{x}_{Calib}^\top & t_d \end{bmatrix}^\top, \tag{3.3}$$

where $t_d = t_C - t_I$ denotes the time-offset between the camera clock and the IMU clock, which treats the IMU clock as the true time. The state vector of IMU at time step k can

be written as:

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_k}^\top & {}^{G}\mathbf{v}_{I_k}^\top & \mathbf{b}_{g_k}^\top & \mathbf{b}_{a_k}^\top \end{bmatrix}^\top, \tag{3.4}$$

where ${}^{I_k}_{G}\bar{q}$ denotes the JPL unit quaternion from the global frame to the IMU frame. ${}^{G}\mathbf{p}_{I_k}$ and ${}^{G}\mathbf{v}_{I_k}$ are the position and velocity of IMU in the global frame. $\mathbf{b}_{g_k}$ and $\mathbf{b}_{a_k}$ represent the gyroscope and accelerometer biases. During inference, the robot maintains a sliding window with m cloned IMU poses at time step k written as:

$$\mathbf{x}_{Clone_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^{-1\top} & {}^{G}\mathbf{p}_{I_{k-1}}^\top & \cdots & {}^{I_{k-m}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-m}}^\top \end{bmatrix}^\top. \tag{3.5}$$

In addition to the IMU state, the historical SLAM features are also stored in the state vector as:

$$\mathbf{x}_f = \begin{bmatrix} {}^{G}\mathbf{p}_{f_1}^\top & \cdots & {}^{G}\mathbf{p}_{f_i}^\top \end{bmatrix}^\top, \tag{3.6}$$

and spatial calibration between its IMU frame and camera frame will also be estimated as:

$$\mathbf{x}_{Calib_k} = \begin{bmatrix} {}^{C_k}_{I_k}\bar{q}^\top & {}^{C_k}\mathbf{p}_{I_k}^\top \end{bmatrix}^\top. \tag{3.7}$$

3.3.2 IMU Dynamic Model

The measurement of the IMU linear acceleration ${}^{I}\mathbf{a}_m$ and the angular velocity ${}^{I}\boldsymbol{\omega}_m$ are modeled as:

$${}^{I}\mathbf{a}_m = {}^{I}\mathbf{a} + {}^{I}_{G}\mathbf{R}\,{}^{G}\mathbf{g} + \mathbf{b}_a + \mathbf{n}_a, \tag{3.8}$$

$${}^{I}\boldsymbol{\omega}_m = {}^{I}\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g, \tag{3.9}$$

where ${}^{I}\mathbf{a}$ and ${}^{I}\boldsymbol{\omega}$ are the true linear acceleration and angular velocity. $\mathbf{n}_a$ and $\mathbf{n}_g$ represent the continuous-time Gaussian noises that contaminate the IMU measurements. ${}^{G}\mathbf{g}$ denotes the gravity expressed in the global frame. Then, the dynamic system of IMU can be modeled the same as in Section 2.3.2.

3.3.3 Initialization Model

The purpose of the initialization model is to restore the IMU state at the first timestamp $\mathbf{x}_{I_0}$ from a prior map $\mathcal{N}_\theta$. In iNeRF[98], a 6 Degrees of Freedom (DoF) pose estimation is proposed, leveraging gradient descent to reduce the residual between pixels generated from a NeRF and those within an observed image. However, this approach heavily depends on a good initial guess.

In this section, we introduce a novel MLP-based initialization model that directly maps images to poses without needing an initial estimate. Specifically, we pre-collect images and groundtruth data from the same environment and train a 7-layer MLP. This MLP encodes prior environmental knowledge, taking a sequence of images as input and outputting 6-DoF poses. Working with pose estimation in the context of $\mathfrak{se}(3)$ requires careful consideration of the underlying Lie group structure. The lack of invariance in the standard inner product on $\mathfrak{se}(3)$ has a potential drawback, as it can lead to discrepancies when comparing poses in different coordinate frames. Hence, our contribution goes beyond just initialization, as we define our loss function using geodesic distance on $SE(3)$ with a left-invariant metric. This ensures consistent and invariant pose comparisons, addressing the limitations tied to inner product-based metrics.

In the establishment of a left-invariant metric on $SE(3)$, the definition involves specifying the inner product on the Lie algebra $\mathfrak{se}(3)$ and subsequently extending it to a Riemannian metric through left translation[72]. The left-invariant metric is established when the following equation holds[37]:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}} = \left\langle \mathbf{S}^{-1}\mathbf{x}_1, \mathbf{S}^{-1}\mathbf{x}_2 \right\rangle_{\mathbf{I}}, \tag{3.10}$$

where $\langle \cdot, \cdot \rangle_{\mathbf{S}}$ represents the inner product within the tangent space $T_{\mathbf{S}}SE(3)$ at an arbitrary element $\mathbf{S} \in SE(3)$, $\mathbf{x}_1, \mathbf{x}_2 \in T_{\mathbf{S}}SE(3)$, $\mathbf{I}$ denotes the identity, and $(\cdot)^{-1}$ denotes the inverse operation in the Lie group $SE(3)$.

Inspired by the definition of bi-invariant metric in $SO(3)$, the metric in $SE(3)$ can be constructed similarly. We define

$$\mathbf{M}_{\mathfrak{se}(3)} = \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{a} \\ \mathbf{a}^T & 1 \end{bmatrix},$$

where $\mathbf{a} \in \mathbb{R}^3$. The eigenvalues of $\mathbf{M}_{\mathfrak{se}(3)}$ are $1, 1 \pm \|\mathbf{a}\|_2$, and the condition $\|\mathbf{a}\|_2 < 1$ ensures all eigenvalues are positive. Then, the metric on $T_{\mathbf{S}}SE(3)$ is defined as:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}} = \mathrm{tr}(\mathbf{x}_1^T \mathbf{x}_2 \mathbf{M}_{\mathfrak{se}(3)}) \cdot \tag{3.11}$$

Lemma 1 *Left-invariant: The metric defined in (3.11) is left-invariant.*

Proof. For $\mathbf{S} = \begin{bmatrix} \mathbf{R}_s & \mathbf{p}_s \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$, let $\mathbf{x}_i \in T_{\mathbf{S}}SE(3)$, $i = \{1, 2\}$, be $\mathbf{x}_i = \begin{bmatrix} \lfloor \boldsymbol{\omega}_{i,s} \times \rfloor & \mathbf{v}_{i,s} \\ \mathbf{0} & 0 \end{bmatrix}$,

we have

$$\mathbf{S}^{-1}\mathbf{x}_i = \begin{bmatrix} \mathbf{R}_s^T & -\mathbf{R}_s^T\mathbf{p}_s \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \lfloor \boldsymbol{\omega}_{i,s}\times \rfloor & \mathbf{v}_{i,s} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_s^T\lfloor \boldsymbol{\omega}_{i,s}\times \rfloor & \mathbf{R}_s^T\mathbf{v}_{i,s} \\ \mathbf{0} & 0 \end{bmatrix}.$$

Then, according to (3.11), we have

$$\begin{aligned} &\left\langle \mathbf{S}^{-1}\mathbf{x}_1, \mathbf{S}^{-1}\mathbf{x}_2 \right\rangle_{\mathbf{I}} \\ &= \mathrm{tr}\left(\begin{bmatrix} \mathbf{R}_s^T\lfloor \boldsymbol{\omega}_{1,s}\times \rfloor & \mathbf{R}_s^T\mathbf{v}_{1,s} \\ \mathbf{0} & 0 \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_s^T\lfloor \boldsymbol{\omega}_{2,s}\times \rfloor & \mathbf{R}_s^T\mathbf{v}_{2,s} \\ \mathbf{0} & 0 \end{bmatrix} \mathbf{M}_{\mathfrak{se}(3)} \right) \\ &= \mathrm{tr}\left(\begin{bmatrix} \lfloor \boldsymbol{\omega}_{1,s}\times \rfloor^T\lfloor \boldsymbol{\omega}_{2,s}\times \rfloor & \lfloor \boldsymbol{\omega}_{1,s}\times \rfloor^T\mathbf{v}_{2,s} \\ \mathbf{v}_{1,s}^T\lfloor \boldsymbol{\omega}_{2,s}\times \rfloor & \mathbf{v}_{1,s}^T\mathbf{v}_{2,s} \end{bmatrix} \mathbf{M}_{\mathfrak{se}(3)} \right) \\ &= \mathrm{tr}\left(\begin{bmatrix} \lfloor \boldsymbol{\omega}_{1,s}\times \rfloor & \mathbf{v}_{1,s} \\ \mathbf{0} & 0 \end{bmatrix}^T \begin{bmatrix} \lfloor \boldsymbol{\omega}_{2,s}\times \rfloor & \mathbf{v}_{2,s} \\ \mathbf{0} & 0 \end{bmatrix} \mathbf{M}_{\mathfrak{se}(3)} \right) \\ &= \left\langle \mathbf{x}_1, \mathbf{x}_2 \right\rangle_{\mathbf{S}}, \end{aligned}$$

which means that the metric is left-invariant. ■

We denote f_1, f_2 as the corresponding local flows with

$$\mathbf{x}_1 = \dot{f}_1(t) \quad \mathbf{x}_2 = \dot{f}_2(t) \quad f_1(t) = f_2(t) = \mathbf{S}. \tag{3.12}$$

As $f_i(t) \in SE(3)$, it can be written as:

$$f_i(t) = \begin{bmatrix} \mathbf{R}_i(t) & \mathbf{p}_i(t) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.13}$$

and the corresponding twists at time t can be expressed as :

$$\mathbf{T}_i = f_i^{-1}(t)\dot{f}_i(t) = \begin{bmatrix} \lfloor \boldsymbol{\omega}_i \times \rfloor & \mathbf{v}_i \\ \mathbf{0} & 0 \end{bmatrix}. \tag{3.14}$$

With the definition of the metric in (3.11), the inner product can be reformulated as:

$$\begin{aligned} \langle \mathbf{x}_1, \mathbf{x}_2 \rangle_{\mathbf{S}} &= \mathrm{tr}(\dot{f}_1^T(t)\dot{f}_2(t)\mathbf{M}_{\mathfrak{se}(3)}) \\ &= \mathrm{tr}(\mathbf{T}_1^T\mathbf{T}_2\mathbf{M}_{\mathfrak{se}(3)}) \quad \longleftarrow \text{Left Invariance} \\ &= \mathrm{tr}(\lfloor \boldsymbol{\omega}_1 \times \rfloor^T \lfloor \boldsymbol{\omega}_2 \times \rfloor) + \mathrm{tr}(\lfloor \boldsymbol{\omega}_1 \times \rfloor^T \mathbf{v}_2 \mathbf{a}^T) + \mathbf{v}_1^T \lfloor \boldsymbol{\omega}_2 \times \rfloor \mathbf{a} + \mathbf{v}_1^T \mathbf{v}_2 \\ &= \begin{bmatrix} \boldsymbol{\omega}_1 \\ \mathbf{v}_1 \end{bmatrix}^T \begin{bmatrix} 2\mathbf{I}_{3\times 3} & \lfloor \mathbf{a} \times \rfloor \\ \lfloor -\mathbf{a} \times \rfloor & \mathbf{I}_{3\times 3} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_2 \\ \mathbf{v}_2 \end{bmatrix} \\ &:= \left\langle \begin{bmatrix} \boldsymbol{\omega}_1 \\ \mathbf{v}_1 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\omega}_2 \\ \mathbf{v}_2 \end{bmatrix} \right\rangle_{\mathbf{M}_{\mathfrak{se}(3)}}. \end{aligned} \tag{3.15}$$

The left-invariant metric on $\mathfrak{se}(3)$ allows us to define the geodesic loss on $SE(3)$ as follows:

$$\begin{aligned} d^2(\mathbf{S}_1, \mathbf{S}_2) &= \left\langle \log_{\mathbf{S}_1}(\mathbf{S}_2), \log_{\mathbf{S}_1}(\mathbf{S}_2) \right\rangle_{\mathbf{S}_1} \\ &= \left\langle \begin{bmatrix} \lfloor \boldsymbol{\omega} \times \rfloor & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix}, \begin{bmatrix} \lfloor \boldsymbol{\omega} \times \rfloor & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix} \right\rangle_{\mathbf{S}_1} \\ &= \left\langle \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} \right\rangle_{\mathbf{M}_{\mathfrak{se}(3)}}, \end{aligned} \tag{3.16}$$

where $\mathbf{S}_1, \mathbf{S}_2 \in SE(3)$, $\log_{\mathbf{S}_1}(\cdot)$ represents Lie logarithm at $\mathbf{S}_1$, $\boldsymbol{\omega}$ and $\mathbf{v}$ denote the rotational velocity and translational velocity from $\mathbf{S}_2$ to $\mathbf{S}_1$, respectively. Since the original data naturally lies in $\mathfrak{se}(3)$, this metric formulation offers computational advantages over the standard left-invariant metric on $SE(3)$. Specifically, it eliminates the need for mapping between $\mathfrak{se}(3)$ and $SE(3)$, while maintaining mathematical rigor through the use of the canonical inner product structure on $\mathfrak{se}(3)$. This approach both simplifies computation and preserves the geometric interpretation of the distance measure.

With this loss function, we train an MLP-based initialization model to relocalize the first captured image in the prior map $\mathbf{T}_W^{C_0} := \left[{}_W^{C_0}\bar{q}^\top, {}^{C_0}\mathbf{p}_W^\top\right]$. Based on the IMU integration in (2.16) and (2.17), and the calibration parameters in (3.7), the relative transformation from the first IMU pose to the first camera pose $\mathbf{T}_{C_0}^{I_0} := \left[{}_{C_0}^{I_0}\bar{q}^\top, {}^{I_0}\mathbf{p}_{C_0}^\top\right]$ can be obtained. As shown in Figure 3.3, the first IMU frame can be relocalized in the prior map frame as:

$$\mathbf{T}_W^{I_0} = \mathbf{T}_{C_0}^{I_0} * \mathbf{T}_W^{C_0}. \tag{3.17}$$

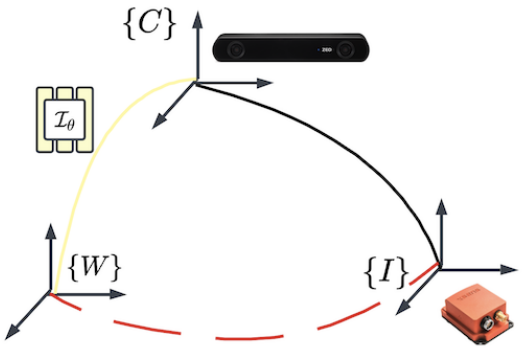


Figure 3.3: IMU pose initialization. From the init model, the relative pose between the first camera frame and the prior map frame can be determined. With the camera-IMU calibration parameters and the timestamps, the transformation between the first camera frame and the first IMU frame can be found.

To further initialize $\left[{}^{G}\mathbf{v}_{I_0}^{\top},\, \mathbf{b}_{g0}^{\top},\, \mathbf{b}_{a0}^{\top}\right]$, we collect a window of IMU readings from timestamp 0 to the time received the first image, and initialize using the average of velocities and bias within this window.

3.3.4 Robustness to Environmental Alterations

To address dynamic objects and minor environmental alterations between the previous map and the current scenario, we introduce the grid-based Structural Similarity Index (SSIM) [92] algorithm. This method involves partitioning both the captured and rendered images into numerous small grids and computing the SSIM similarity for each grid pair across the two images as:

$$\mathrm{SSIM}\left(I_x, I_y\right) = \left[l(I_x, I_y)\right]^{\alpha} \cdot \left[c(I_x, I_y)\right]^{\beta} \cdot \left[s(I_x, I_y)\right]^{\gamma}, \tag{3.18}$$

where $l(I_x, I_y)$, $c(I_x, I_y)$, and $s(I_x, I_y)$ denote the local mean (luminance), standard deviations (contrast), and cross-covariance (structural) similarity between two images. α, β, and γ denote the weights for three terms. If the similarity of a grid pair surpasses a predetermined threshold, feature extraction will be applied to both grids. Conversely, regions where the similarity falls below the threshold are deemed to contain dynamic objects, and consequently, no feature will be extracted within those areas on the rendered image.

3.3.5 Measurement Update using Captured Images

The feature measurements captured from an onboard camera can be described by:

$$\mathbf{z}_c = \Pi\left({}^C\mathbf{p}_f\right) + \mathbf{w}_c, \quad \Pi\left([x\ y\ z]^\top\right) = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^\top, \tag{3.19}$$

where ${}^C\mathbf{p}_f$ denotes the position of this feature in the camera frame, and $\mathbf{w}_c$ denotes the corresponding measurement noise. Based on the estimated relative transformation between IMU and the global frame, the estimated calibration parameters, ${}^C\mathbf{p}_f$ can be expressed as:

$${}^C\mathbf{p}_f = {}^C_I\mathbf{R}\,{}^I_G\mathbf{R}\left(\bar{t}\right)\left({}^G\mathbf{p}_f - {}^G\mathbf{p}_I\left(\bar{t}\right)\right) + {}^C\mathbf{p}_I, \tag{3.20}$$

where $\bar{t} = t - t_d$ is the exact camera time between the global and IMU frame.

To update a particular captured feature, we first gather all measurements of this feature within the current sliding window. Then, the measurement residuals are computed between each observation and the registered feature. By stacking all measurement residuals,

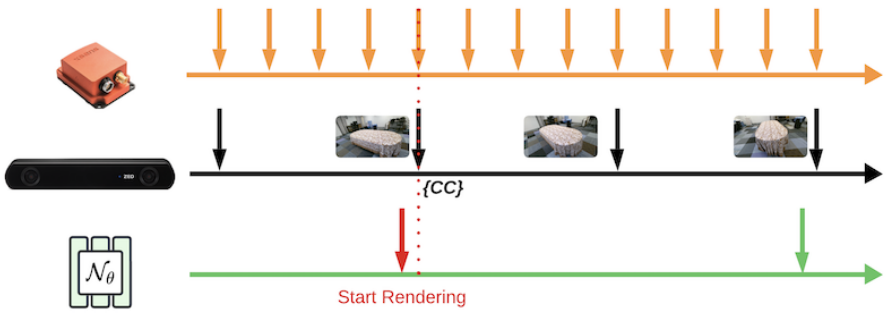


Figure 3.4: The three timelines denote data received from different sensors and the NeRF model. We define the closest camera frame $\{CC\}$ as the frame closest in time to when the NeRF model begins rendering.

we linearize them at the estimated IMU pose as follows:

$$\mathbf{r}_c = \mathbf{h}_c \left(\tilde{\mathbf{x}}, {}^G\tilde{\mathbf{p}}_f \right) + \mathbf{w}_c \simeq \mathbf{H}_{x,c}\tilde{\mathbf{x}} + \mathbf{H}_{f,c}{}^G\tilde{\mathbf{x}}_f + \mathbf{w}_c, \tag{3.21}$$

where $\mathbf{H}_{x,c}$ and $\mathbf{H}_{f,c}$ denote the state and measurement Jacobians of captured features, respectively. $\mathbf{w}_c$ denotes the noise vector corresponding to the captured feature. Then, the standard MSCKF update [65] is applied using left-nullspace projection for $\mathbf{H}_f$.

3.3.6 Measurement Update using Rendered Images

To incorporate the observations from the rendered image and update the state vector, we aim to update the state corresponding to the pose at which the image was rendered. However, due to factors such as rendering delay and the fact that the camera pose has already been updated based on captured features, we opt for the closest camera frame $\{CC\}$ relative to the original rendered one as shown in Fig. 3.4. The measurement

function of rendered features can be formulated as:

$$\mathbf{z}_r = \Pi\left({}^{CC}\mathbf{p}_f\right) + \mathbf{w}_r, \tag{3.22}$$

where $\mathbf{w}_r$ denotes the rendered noise, and

$${}^{CC}\mathbf{p}_f = {}^{CC}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}\left(\bar{t}\right)\left(\left({}^{G}_{W}\mathbf{R}\,{}^{W}\mathbf{p}_f + {}^{G}\mathbf{p}_W\right) - {}^{G}\mathbf{p}_I\left(\bar{t}\right)\right) + {}^{C}\mathbf{p}_I. \tag{3.23}$$

The error state Jacobians w.r.t. the pose of IMU can be expressed as:

$$\begin{aligned} \frac{\partial \tilde{\mathbf{z}}_r}{\partial \delta^{I}_{G}\theta} &= \mathbf{J}_{\Pi}\,{}^{CC}_{I}\mathbf{R}\lfloor\,{}^{I}_{G}\mathbf{R}\left(\bar{t}\right)\left({}^{G}_{W}\mathbf{R}\,{}^{W}\mathbf{p}_f + {}^{G}\mathbf{p}_W - {}^{G}\mathbf{p}_I\left(\bar{t}\right)\right)\times\rfloor, \\ \frac{\partial \tilde{\mathbf{z}}_r}{\partial {}^{G}\tilde{\mathbf{p}}_I} &= -\mathbf{J}_{\Pi}\,{}^{CC}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}\left(\bar{t}\right), \end{aligned} \tag{3.24}$$

where $\mathbf{J}_{\Pi}$ denotes the Jacobian of perspective model.

Note that the rigid transformation $\left({}^{G}_{W}\mathbf{R}, {}^{G}\mathbf{p}_W\right)$ from the initialization model is not perfect, but (3.23) has not modeled the initialization noise into it. Thus, we inflate the noise term as:

$$\mathbf{w}'_r = \mathbf{w}_r + \frac{\partial \mathbf{z}_r}{\partial \delta^{G}_{W}\theta} * {}^{G}_{W}\tilde{\theta} + \frac{\partial \mathbf{z}_r}{\partial {}^{G}\tilde{\mathbf{p}}_W} * {}^{G}\tilde{\mathbf{p}}_W, \tag{3.25}$$

where

$$\begin{aligned} \frac{\partial \mathbf{z}_r}{\partial {}^{G}_{W}\mathbf{R}} &= \mathbf{J}_{\Pi}\,{}^{CC}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}\lfloor\,{}^{G}_{W}\mathbf{R}\,{}^{W}\mathbf{p}_f\times\rfloor, \\ \frac{\partial \mathbf{z}_r}{\partial {}^{G}\mathbf{p}_W} &= \mathbf{J}_{\Pi}\,{}^{CC}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R}\left(\bar{t}\right). \end{aligned} \tag{3.26}$$

Then, the linearized model can be expressed as:

$$\mathbf{r}_r = \mathbf{h}_r\left(\tilde{\mathbf{x}}, {}^W\tilde{\mathbf{p}}_f\right) + \mathbf{w}'_r \simeq \mathbf{H}_{x,r}\tilde{\mathbf{x}} + \mathbf{H}_{f,r}{}^W\tilde{\mathbf{x}}_f + \mathbf{w}'_r, \tag{3.27}$$

and an EKF update [65] will be employed.

3.4 Experiments

In this section, we validate the performance of NeRF-VIO initialization and localization using a real-world AR table dataset [29]. The dataset comprises AR table sequences 1-4, recorded around a table adorned with a textured tablecloth. Table sequence 5 introduces minor alterations by incorporating additional objects onto the table (minor environment change), while table sequences 6-8 place an additional large whiteboard to simulate the large environment change. Throughout the training process, sequence 1 is utilized to train both the initialization model and the NeRF model on an RTX 4090 GPU. In Sec. 3.4.1, we compare the accuracy and latency of initialization with iNeRF [98]. Rendering performance and VIO localization accuracy are evaluated and compared with MSCKF [65] in Sec. 3.4.2. Additionally, Sec. 3.4.3 showcases an instance of grid-based SSIM.

3.4.1 Initialization Performance

The initialization model is trained as a 7-layer MLP using AR table sequence 1. RGB images are extracted from a Rosbag, which records from an Intel RealSense D455 camera [3]. The IMU groundtruth are captured from the Vicon system [5], and camera intrinsic and camera-IMU extrinsic parameters are calibrated using Kalibr [77]. Before for-

warding the images to MLP, the corresponding camera poses are determined using 4th-order Runge-Kutta interpolation [25]. RGB images are then converted to grayscale, normalized to a range between 0 and 1, and processed through the MLP.

To compare our initialization model with iNeRF, we leverage our pre-trained NeRF model from NeRF-PyTorch[1] as a prior map. Pose estimation of the first images in sequences 2-8 is conducted using iNeRF[2] and our initialization model. Specifically, we initialized iNeRF with two different initial guesses: **(a)** a 10-degree rotational error and a 20-centimeter translation error for each axis. **(b)** a 2-degree rotational error and a 5-centimeter translation error for each axis. We evaluate the L_2 norm of position and orientation between estimated values and groundtruth of those two models in Table 3.1, while latency is provided in Table 3.2. We can figure out that our NeRF-VIO initialization model demonstrates superior performance over iNeRF across all four trajectories, exhibiting significantly lower latency. This can be attributed to iNeRF's reliance on gradient-based optimization, which needs to converge to local minima iteratively. Notable that we preload all models before initialization, thus the model loading time is not contained in Table 3.2. Additionally, iNeRF relies on a NeRF prior map, which renders it vulnerable to significant environmental changes, leading to relocalization failures as observed in Table 5. In contrast, our model exhibits robustness to minor environmental alterations and retains the capability to reconstruct images even when a large environment changes.

[1] `https://github.com/yenchenlin/nerf-pytorch`.
[2] `https://github.com/salykovaa/inerf`.

Table 3.1: The L_2 norm of the orientation / position (degrees / centimeters) of the initialization pose, utilizing iNeRF and our NeRF-VIO across AR table sequences 2-8. For iNeRF, we use different initial guesses: **(a)** a 10-degree rotational error and a 20-centimeter translation error for each axis. **(b)** a 2-degree rotational error and a 5-centimeter translation error for each axis.

	iNeRF (a)	**iNeRF (b)**	**NeRF-VIO**
Table 2	20.33 / 23.39	2.81 / 5.49	2.02 / 1.48
Table 3	9.95 / 38.37	2.70 / 4.79	2.71 / 2.04
Table 4	10.61 / 22.95	3.35 / 6.55	3.16 / 1.90
Table 5	-	5.47 / 8.09	5.21 / 4.76
Table 6	-	-	4.61 / 2.30
Table 7	-	-	5.28 / 2.67
Table 8	-	-	4.78 / 1.27

Table 3.2: The latency (seconds) of pose generation, utilizing iNeRF and our NeRF-VIO across AR table sequences 2-8.

	Table 2	**Table 3**	**Table 4**	**Table 5**	**Table 6**	**Table 7**	**Table 8**
iNeRF	15.46	15.55	15.64	-	-	-	-
NeRF-VIO	0.11	0.12	0.13	0.11	0.10	0.12	0.11

3.4.2 VIO Performance

The NeRF model is constructed with 8 fully connected layers, followed by concatenation with the viewing direction of the camera, and passed through an additional fully connected layer. In addition to the image preprocessing outlined in Sec. 3.4.1, we further rotate the camera by 180 degrees along the x-axis to maintain consistent rendering direction. Fig. 3.5 illustrates the testing results of the NeRF model at various iterations during training. To evaluate the capability of NeRF-VIO to adapt to small or large environmental changes, a comparison of rendered images and ground truth is presented in Fig. 3.6, utilizing data from AR Table 3-6.

In assessing the impact of rendered image updates and initialization models on VIO performance, we compare our NeRF-VIO with MSCKF [65] and NeRF-VIO (GT Init),

Figure 3.5: Testing results of NeRF model. From left to right, the images represent the groundtruth of the test image, the rendered image at iteration 1,000, the rendered image at iteration 50,000, and the rendered image at iteration 200,000.

Figure 3.6: Comparison of NeRF-rendered images to ground truth under normal / minor-change / large-change environments. The top row displays captured images from the closest camera frame, while the second row showcases rendered images at the same positions and orientations. Columns correspond to Table 3-6, progressing from left to right.

which same as NeRF-VIO but initialized from ground truth. To ensure a fair comparison, we employ the same seed and an equal number of KLT features [60] for all three methods. For NeRF-VIO, we run the NeRF rendering at 2Hz and the onboard camera at 30Hz on an Intel 9700K CPU. Table. 3.3 presents the absolute trajectory error (ATE) from Table 2-8, while Fig. 3.7 displays the relative pose error (RPE) of AR Table 4. It is evident that our NeRF-VIO outperforms MSCKF across all sequences and achieves performance nearly on par with the groundtruth initialization.

Table 3.3: The ATE of the orientation / position (degrees / meters) of three VIO methods in different AR Table sequences 2-8. For NeRF-VIO, Table 1 is trained and used as a prior map for all sequences.

	MSCKF	**NeRF-VIO**	**NeRF-VIO (GT Init)**
Table 2	1.142 / 0.034	**0.686** / **0.023**	0.750 / 0.024
Table 3	0.750 / 0.065	0.651 / 0.049	**0.517** / **0.046**
Table 4	2.095 / 0.077	0.886 / **0.038**	**0.766** / 0.040
Table 5	0.656 / 0.047	**0.519** / **0.028**	0.534 / 0.031
Table 6	0.961 / 0.049	0.737 / 0.036	**0.564** / **0.028**
Table 7	1.161 / 0.069	0.982 / 0.049	**0.896** / **0.043**
Table 8	1.319 / 0.063	0.963 / 0.038	**0.881** / **0.025**
Average	1.155 / 0.058	0.775 / 0.037	**0.701** / **0.034**

3.4.3 Robust to Environment Changes

To further classify the mechanics of the grid-based SSIM, we provide an example in Fig. 3.8 to illustrate both pixel-level and grid-level similarity. Contrasting with the third column of Fig. 3.6, the presence of dark pixels in Fig. 3.8(a) signifies a high similarity computed between the rendered and captured images. In our implementation, we assign a weight of $[1, 0.5, 0.1]$ to the exponent term in (3.18), and the SSIM for each grid is shown in Fig. 3.8(b). Then, only grids exhibiting a similarity that is larger than 0.8 are utilized for FAST [90] feature extraction. This methodology ensures consistency between the NeRF map and the real map while maintaining robustness against environmental changes.

3.5 Conclusions

In this chapter, we have proposed a map-based visual-inertial odometry algorithm with pose initialization. We define a novel loss function for the initialization model and train an MLP model to recover the pose from a prior map. Besides, we proposed a two-stage update pipeline integrated into the MSCKF framework. Through the evaluation on

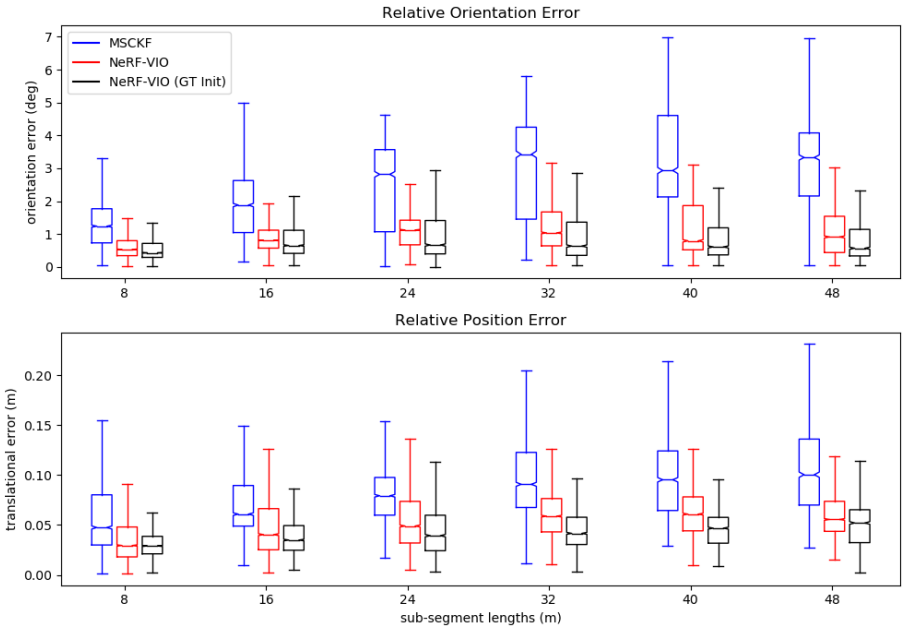


Figure 3.7: The RPE of MSCKF [65], NeRF-VIO (ours), and NeRF-VIO (GT Init) using AR Table 4. NeRF-VIO initializes from the pre-trained model, while NeRF-VIO (GT Init) initializes directly from groundtruth.

a real-world AR dataset, we demonstrate that our NeRF-VIO outperforms all baselines in terms of both accuracy and efficiency.

(a) Pixel-level Similarity map

(b) Grid-level Similarity map

Figure 3.8: Comparison of pixel-level and grid-based SSIM. (a) A dark region denotes a high similarity, while the white region denotes a huge luminance, contrast, and structural difference weighted by $[1, 0.5, 0.1]$. (b) A grid-level similarity map is used in our algorithm. The red text denotes the similarity of each small grid.

Chapter 4

Infrastructure-less Cooperative SLAM for Multi-user Augmented Reality

4.1 Introduction and Related Works

Immersive technologies have facilitated the development of numerous successful mobile applications offering users AR experiences. These AR applications encompass a wide range of use cases, much as mobile games [13], navigation apps [11, 22], in-vehicle AR systems [8, 10], and social AR apps [14]. In these multi-user AR applications, multiple users can simultaneously place, view, manipulate, and interact with shared virtual objects.

A fundamental requirement for enabling such interactions is precise six degree-of-freedom (6DoF) localization within a common coordinate system. Collaborative simultane-

ous localization and mapping (C-SLAM) is a widely adopted approach to achieve this. By employing C-SLAM, AR frameworks establish a shared reference frame among users, ensuring that virtual objects appear correctly aligned across different perspectives. For instance, some systems host a cloud server to maintain a global map, such that users can either upload their local observations and get localization results [19], or keep a local map and upload partial maps periodically for stitching and alignment [81, 15, 95]. However, these approaches require a centralized edge/cloud server and assume stable connections from the users to the server, as the users transmit a significant volume of data to the server for mapping and localization. Instead of a full-fledged map, one can also define sparse cloud-hosted anchors to position user headsets. These anchors are lightweight but still require a centralized host as a bridge to connect users.

While centralized infrastructure offloads intensive computational tasks and facilitates user interaction, it is not always viable in scenarios like search-and-rescue operations in remote areas, firefighting missions, or ad hoc gaming setups [16]. In such cases, infrastructure-free solutions become essential to broaden AR applicability. Apple's ARKit [7], for example, employs a peer-to-peer (P2P) approach, synchronizing maps directly between users for localization. However, this synchronization introduces delays and significant data transmission during initialization. Moreover, it restricts interactions with users with pre-synchronized maps, requiring new users to synchronize before joining the shared AR session. Furthermore, the approach confines interactions to pre-synchronized areas, limiting AR experiences in newly explored environments that emerge after synchronization.

To achieve the above design goals, there are three significant challenges that have to be addressed. First, because there is no infrastructure support, AR users need to data sharing amongst themselves using P2P links in an ad-hoc fashion. Intermittent P2P links may be disconnected frequently, while the required data transfers for AR are relatively bulky. Thus, a key challenge is to identify opportunities for sharing and enabling AR visualizations. Instead of constantly uploading local maps to a server, users should share local maps only when there are peer users nearby, to help themselves localize and interact with each other and virtual objects. Second, transmitting and sharing bulky partial local maps over these intermittent connections may result in packet loss and re-transmissions. This causes a delay in the initiation of the AR experience, which may introduce inconsistent visual effects (such as drifting virtual objects) that break the immersive experience. Finally, as the users explore and encounter each other, the map and localization errors may be compounded, such as if small errors build on each other. The problem may be further exacerbated if multiple users have inconsistent local maps inherited from previous encounters.

To this end, we introduce CooperSLAM, a multi-agent, infrastructure-free SLAM framework designed for seamless multi-user AR interactions. Unlike the PL-CVIO and NeRF-VIO as discussed in Chapter 2 and Chapter 3, CooperSLAM prioritizes data compression, efficient transmission, and rapid map alignment among multiple agents. To facilitate lightweight and fast data exchanges, CooperSLAM extracts key frames, eliminates unnecessary complex data structures for localization, and shares only sparse features in compact messages. As users encounter one another, CooperSLAM shares more sparse features and refines the alignment. The collaborative mapping and refinement progress further

as users encounter more peers while traversing a larger spatial area.

Our evaluations demonstrate that CooperSLAM achieves an average of 13.4 cm translation error within 10 meters from the position where users encountered each other, which is 12.9%-37.9% better compared to the *state-of-the-art* centralized C-SLAM approaches. In the meantime, CooperSLAM only transmits 13 KB per frame, and in total 106 KB to successfully achieve map alignment, which is 71.7%-92.4% less data transmitted per localized user. Furthermore, it takes the shortest time to achieve the map alignment; the delay is up to 65% shorter than achievable with the baselines.

4.2 Preliminaries

4.2.1 Pose Graph Optimization

A pose graph [85, 44] is a structured representation used in SLAM to model the relationships between different states of a robot and its surrounding environment. It consists of two primary components: *nodes* and *edges* as shown in Figure 4.1.

The nodes represent discrete states of the robot over time, capturing its position, orientation, and potentially other relevant parameters. Additionally, nodes can also correspond to environmental landmarks, which serve as reference points for localization and mapping. The edges define the constraints between these nodes, which are derived from various sensor measurements. In this context, the edges are typically formed based on data collected from IMU and camera observations. The IMU provides motion constraints by measuring acceleration and angular velocity, while the camera captures visual features that help in estimating relative transformations between different poses. By incorporating these

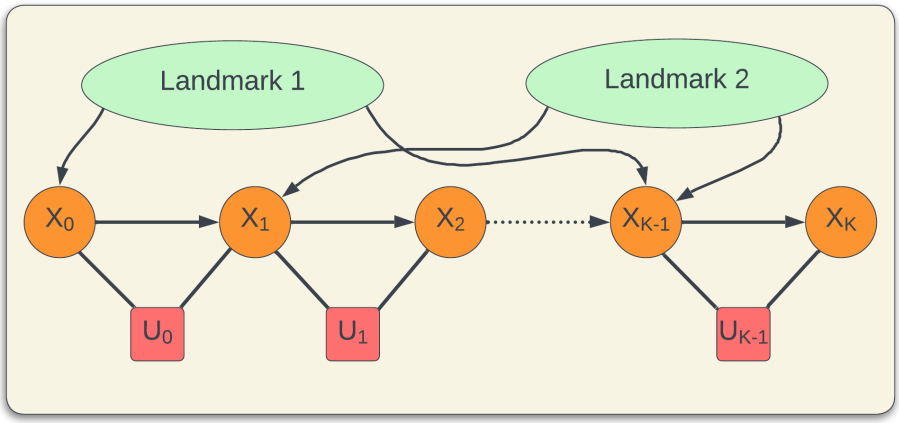


Figure 4.1: An example of a pose graph: orange and green circles represent the robot state and landmark nodes, while red squares indicate the IMU measurement constraints.

measurement constraints, a pose graph enables optimization to refine the estimated pose of the robot and improve mapping accuracy.

4.2.2 Gauss-Newton and Levenberg–Marquardt Algorithms

To optimize a pose graph in VIO/SLAM, the problem is typically formulated as a nonlinear least squares optimization. The objective function can be written as:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \Rightarrow f(\mathbf{x}) &= \underset{\mathbf{x}\in\mathbb{R}^n}{\text{Minimize}} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2, \end{aligned} \tag{4.1}$$

where $\mathbf{A}$ represents the linearized Jacobian matrix, $\mathbf{x}$ is the state vector to be optimized, and $\mathbf{A}\mathbf{x}-\mathbf{b}$ represents the residual error at the current estimate for a given feature. Since the system is often overdetermined, instead of using a direct closed-form (least square) solution, iterative optimization methods such as Gauss-Newton and Levenberg-Marquardt are commonly employed.

The Gauss-Newton method is an approximation of Newton's method [54] that is widely used for nonlinear least squares problems. While it converges more slowly than the full Newton's method, it is more robust, especially when handling noisy or ill-conditioned data. The Gauss-Newton approach minimizes the objective function by iteratively solving:

$$\begin{aligned} f(\mathbf{x}+\Delta\mathbf{x}) &\simeq \|f(\mathbf{x})+J(\mathbf{x})\Delta\mathbf{x}\|_2^2 \\ &= \Delta\mathbf{x}^\top J^\top(\mathbf{x})J(\mathbf{x})\Delta\mathbf{x} + 2f^\top(\mathbf{x})J(\mathbf{x})\Delta\mathbf{x} + f^\top(\mathbf{x})f(\mathbf{x}) \end{aligned} \tag{4.2}$$

where $f^\top(\mathbf{x})J(\mathbf{x}) = J^\top(\mathbf{x})f(\mathbf{x})$, and $J(\mathbf{x})$ denotes the Jacobian matrix evaluated at the current estimate $\mathbf{x}$. Setting the derivative of the cost function with respect to $\Delta\mathbf{x}$ to zero, we obtain the update step:

$$\begin{aligned} & \frac{\partial f(\mathbf{x}+\Delta\mathbf{x})}{\partial\Delta\mathbf{x}} = 0 \\ \Rightarrow \quad & 2J^\top(\mathbf{x})J(\mathbf{x})\Delta\mathbf{x} + 2f^\top(\mathbf{x})J(\mathbf{x}) = 0 \\ \Rightarrow \quad & \Delta\mathbf{x} = -(J^\top(\mathbf{x})J(\mathbf{x}))^{-1}J^\top(\mathbf{x})f(\mathbf{x}). \end{aligned} \tag{4.3}$$

This iterative update refines the estimate $\mathbf{x}$ until convergence, effectively minimizing the residual error.

The Levenberg-Marquardt algorithm is a hybrid approach that combines the Gauss-Newton method (for fast convergence near the solution) with gradient descent (to improve robustness against poor initial estimates and ill-conditioned problems). It modifies the

Gauss-Newton update by introducing a damping factor λ, resulting in the following system:

$$
\begin{aligned}
& J^\top(\mathbf{x})J(\mathbf{x})\Delta\mathbf{x} = -J^\top(\mathbf{x})f(\mathbf{x}) \\
\Rightarrow \quad & (J^\top(\mathbf{x})J(\mathbf{x}) + \lambda I)\Delta\mathbf{x} = -J^\top(\mathbf{x})f(\mathbf{x}) \\
\Rightarrow \quad & \begin{cases} \lambda I\Delta\mathbf{x} = -J^\top(\mathbf{x})f(\mathbf{x}) & \text{if } \lambda \text{ is large} \\ J^\top(\mathbf{x})J(\mathbf{x})\Delta\mathbf{x} = -J^\top(\mathbf{x})f(\mathbf{x}) & \text{if } \lambda \text{ is small} \end{cases}
\end{aligned}
\tag{4.4}
$$

where λ controls the balance between Gauss-Newton and gradient descent. When the solution improves, λ is reduced, making Levenberg-Marquardt behave more like Gauss-Newton. When the solution deteriorates, λ is increased, shifting the algorithm toward gradient descent for better stability. This adaptive approach makes Levenberg-Marquardt a powerful optimization method for nonlinear least squares problems in pose graph optimization, balancing convergence speed and robustness to poor initial conditions.

4.3 Problem Formulation

For multiple AR users to interact with the same virtual objects within the same environment, their local SLAM systems must share a common coordinate frame of reference. This ensures that the virtual objects' positions from one user's perspective can be accurately transformed into the perspectives of others. Unlike the PL-CVIO and NeRF-VIO as discussed in Chapter 2 and Chapter 3, respectively, CooperSLAM focuses on data compression, efficient transmission, and rapid map alignment among multiple agents. Figure 4.2 illustrates the end-to-end architecture of CooperSLAM, which is designed to achieve this objective. Each AR user starts with no prior observations or maps of the environment,

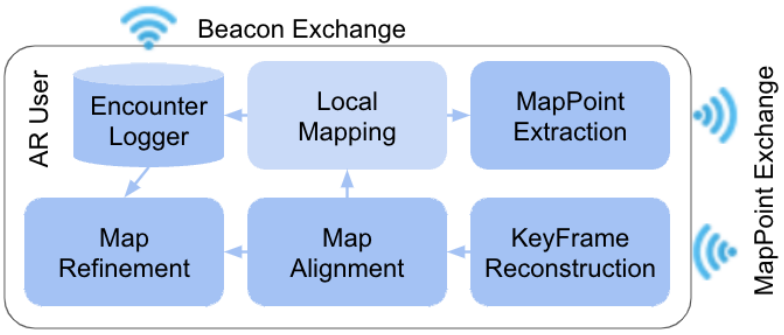


Figure 4.2: CooperSLAM system architecture.

operating from an empty state. Using onboard cameras, users run a local visual SLAM module that performs standard operations, including feature extraction, tracking, feature registration, and map construction. When users encounter each other opportunistically, they exchange discovery beacons as detailed in Section 4.3.2. Upon establishing a connection, CooperSLAM facilitates the exchange of map points between users through these intermittent P2P links. Shadow key frames are then reconstructed using the received map points. These key frames are subsequently utilized for place recognition and map alignment on the device as discussed in Section 4.3.1. The alignment results are fed back into the local mapping and map refinement modules to improve alignment quality. The refinement module in Section 4.3.3 further optimizes the local map by synthesizing multiple key frame poses from both current and previous encounters, as logged in the encounter history.

4.3.1 Lightweight and Robust Map Alignment

A crucial step of C-SLAM is to stitch multiple map segments, which are often generated by the independent local SLAM processes of multiple agents, into a unified map. This process involves aligning these segments within a single coordinate system, enabling

accurate placement and representation of virtual AR objects in the physical environment. To achieve this alignment, the common C-SLAM approach detects co-visible areas and extracts and exchanges observations from different perspectives. Leveraging fine-grained feature matching, C-SLAM systems can then determine the poses where these observations are made by various users in each other's coordinate system. The final alignment is achieved by transforming all users' local maps into a common coordinate system, typically one user's reference frame.

However, a significant challenge arises due to the disparity between the limited communication bandwidth available to mobile users participating in the AR experience and the large volume of map data generated by SLAM processes. Even in cloud or edge-assisted environments with sufficient computational resources, this communication bottleneck prevents real-time map alignment. To address this, existing literature [15, 95] propose compressing map data to reduce its size and reconstructing key frame-based maps at the cloud. Nonetheless, these reduced map representations remain bulky and inflexible. In scenarios with intermittent P2P connectivity, such monolithic data transfers are prone to retransmissions, introduce delays in alignment, and lack the robustness necessary to achieve high map alignment accuracy.

Instead, we completely decompose the map data structure, and transmit the smallest possible feature units, such as individual map points, over the unstable network using a UDP-like approach. *The key insight here is that to achieve reasonable map alignment quality, key frames cannot be lost, but individual map points of a key frame can be partially lost.* This is because as long as a sufficient set of map points remains, map alignment can

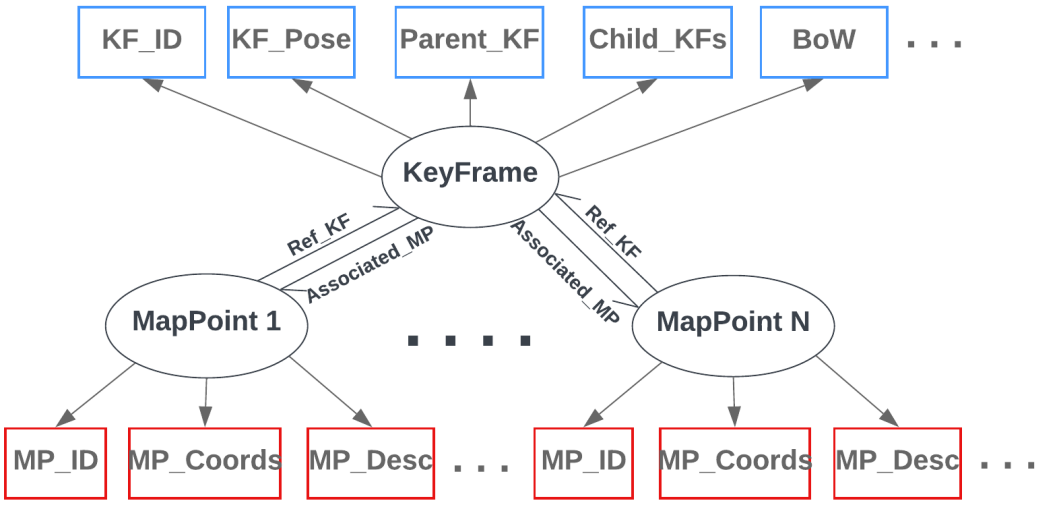


Figure 4.3: Key frame data structure used in CooperSLAM. Each key frame (KF) is associated with multiple map points (MPs), shown as MapPoint 1 to N. Blue rectangles denote the elements that are contained in a key frame, and red rectangles represent map points. The half arrow denotes the pointer between two objects. Other scraped objects are omitted.

still be successful. Figure 4.3 illustrates the key frame data structure of ORB_SLAM2 [68]. Each key frame contains numerous (tens or hundreds of) map points, which collectively can be too large to transmit efficiently. In contrast, a communication message in CooperSLAM is designed as a tiny packet that encapsulates a single map point. Each packet includes the agent ID, key frame ID, map point ID, coordinates, and descriptors, as depicted in Figure 4.4.

While sharing map points offers robustness against packet loss, the challenge lies in reconstructing the map and achieving high alignment accuracy with a potentially reduced set of map points. Existing methods address this by sharing or reconstructing complex

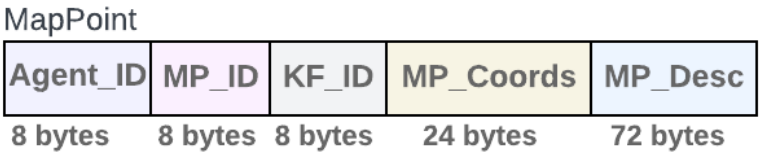


Figure 4.4: MapPoint Message

associated data structures, such as co-visibility graphs, to establish connections between key frames and map points. These methods aim to restore as much data as possible, effectively simulating the scenario where the receiver independently observes and processes all key frames seen by other users using a full-fledged SLAM process.

To resolve the reconstruction complexity with fewer map points, we design a fast initial alignment strategy followed by a robust multi-frame consensus approach. The key observation here is that it does not take all of the associated data structure to accomplish relocalization. Instead, by synthesizing the relocalization results of multiple independent frames, we can achieve similar alignment quality compared to existing approaches.

Shadow Key frame To address this, we introduce the concept of a shadow key frame, an independent key frame reconstructed on the receiver side, which contains only the received map points. In contrast to key frames from SLAM, shadow key frames have two main differences. (1) Due to potential map point loss during transmission, the shadow key frame may not have the complete feature set compared to the corresponding key frame on the sender side. Hence, its Bag of Words (BoW) histogram [38], which is based on the feature set, will also differ. (2) The key frame is reconstructed with no context; therefore it has no awareness of any common features shared with other shadow key frames, and no dependency on or connections to other key frames.

Fast Initial Alignment To minimize map synchronization delay, CooperSLAM achieves faster alignment compared to prior approaches by leveraging a majority of map points from a single key frame to localize it in the local map. Specifically, consider two users, A and B, whose local SLAM creates two coordinate systems, G_A and G_B, respectively. If A receives enough map points from B, it can reconstruct a shadow key frame $\mathcal{K}$ and localize the pose of $\mathcal{K}$ in G_A as $\mathbf{T}_A^{\mathcal{K}}$. Comparing to the estimated pose of the key frame $\mathbf{T}_B^{\mathcal{K}}$ in G_B, user A can align its map using $(\mathbf{T}_B^{\mathcal{K}})^{-1} * \mathbf{T}_A^{\mathcal{K}}$, where $(\cdot)^{-1}$ denotes $SE(3)$ inverse as defined in [88]. Using this alignment, all of user B's key frame poses and the virtual objects' poses in G_B can be transformed into A's coordinate system G_A. For instance, to transform an virtual object O's pose $\mathbf{T}_B^{\mathbf{O}}$ into A's map, the following relationship can be used:

$$\mathbf{T}_A^{\mathbf{O}} = \mathbf{T}_B^{\mathbf{O}} * (\mathbf{T}_B^{\mathcal{K}})^{-1} * \mathbf{T}_A^{\mathcal{K}}. \tag{4.5}$$

Relocalization of the shadow key frame is possible because only the pose of each independent shadow key frame is required. This process involves calculating BoW statistics from the received map points to identify matching candidate frames, followed by fine-grained feature matching to compute the precise transformation.

Alignment Refinement and Outlier Rejection Using map points from only a single key frame can introduce errors. Instead, we propose a consensus multiple key frames stitching algorithm with an outlier rejection strategy as in Algortihm 1. The core idea is to minimize errors in relocalized frames by averaging over multiple frames, and rejecting outlier frames that could skew the results. Specifically, following standard relocalization

Algorithm 1 Alignment Refinement and Outlier Rejection

$\mathcal{K}_{i\in[1:I]} = getAllKeyFrames\,(Map_A)$
$\mathcal{K}_{j\in[1:J]} = getAllKeyFrames\,(Map_B)$
for $i = 1 : I$ **do**
    for $j = 1 : J$ **do**
        $Score = BoW\_Similarity\,(\mathcal{K}_i, \mathcal{K}_j)$
        if $Score \geq t_b$ **then**
            $\mathcal{K}_C \leftarrow \mathcal{K}_C \oplus (\mathcal{K}_i, \mathcal{K}_j)$
        end if
    end for
end for
for $m = 1 : length\,(\mathcal{K}_C)$ **do**
    $F_1, F_2 = Feature\_Extractor\,(\mathcal{K}_C[m, 1], \mathcal{K}_C[m, 2])$
    $nMatches = Matcher\,(F_1, F_2)$
    if $nMatches \geq t_m$ **then**
        $\mathcal{T}_c = Calculate\_TF\,(F_1, F_2) \in SE(3)$
        $\mathcal{T}_C \leftarrow \mathcal{T}_C \oplus \mathcal{T}_c$
    end if
end for
for $n = 1 : length\,(\mathcal{T}_C)$ **do**
    $[x,\ y,\ z,\ ro,\ pi,\ ya] = Log\,(\mathcal{T}_C[n]) \in \mathfrak{se}(3)$
    $X \leftarrow X \oplus x,\ Y \leftarrow Y \oplus y,\ Z \leftarrow Z \oplus z$
    $RO \leftarrow RO \oplus ro,\ PI \leftarrow PI \oplus pi,\ YA \leftarrow YA \oplus ya$
end for
$Outlier\_Reject(X,\ Y,\ Z,\ RO,\ PI,\ YA)$
$\mathcal{T}_A^B = Exp(mean(X,\ Y,\ Z,\ RO,\ PI,\ YA))$
return $\mathcal{T}_A^B \in SE(3)$

procedures, for each reconstructed key frame from received B's map points, we find out the matching key frame candidate from A based on the BoW statistics and the number of matching inlier features. Then, we calculate the transformation matrix between these two key frames using the PnP algorithm [35]. Once we have multiple relocalized key frames between the two users, we convert the transformation matrices to tangent space $\mathfrak{se}(3)$ [88]. To remove the outliers that could be produced when we calculate the transformation matrix candidates, we take all six dimensions of estimates and calculate the mean and variance for each of them. Then, we remove the elements out of the 99% confidential interval for each

dimension of the 6DoF. If any dimension of the current 6-DoF estimate is determined as an outlier, the frame will be removed. This outlier rejection will only be done once, which means the mean and variance will not be recalculated after removing the outliers. Finally, the averaged pose will be transferred back to the transformation matrix on $SE(3)$ [88]. By using this algorithm, we reduce the stitching error and increase the robustness of stitching. At the same time, it can guarantee the orthogonal property of the final rotational matrix part.

4.3.2 Beacon and Alignment Orchestration

In distributed scenarios involving multiple AR users, effective coordination is crucial to scale the pairwise map alignment process to all users in the vicinity. To achieve this, CooperSLAM incorporates a beacon and orchestration mechanism to manage transmissions efficiently.

Periodic beacons are a common method for discovery in mobile ad-hoc networks, and CooperSLAM leverages this approach to establish an implicit handshake before transmitting large volumes of map points. While these transmissions involve many map points, their overall size remains manageable since each map point is individually small. Specifically, when a user receives a beacon from a peer, it signals the potential for mutual communication. CooperSLAM utilizes this opportunity to begin broadcasting the map points of the current key frame, assuming that the receiver may have seen and registered similar map points in its local map. If multiple beacons are received from various agents, the broadcast naturally extends to all users within range, facilitating opportunistic sharing among nearby users.

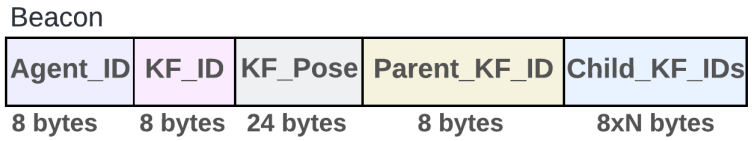


Figure 4.5: Beacon Message

Figure 4.5 shows the composition of the beacon message. Rather than transmitting key frame poses with each map point message, the map point message only includes the associated key frame ID. Beacons are responsible for sending the key frame poses, a task that occurs less frequently than map point transmissions, thereby improving communication efficiency. On the receiver side, the reconstructed shadow key frame is matched to its corresponding key frame in the sender's coordinate frame. The poses of these key frames are then used to calculate the transformation matrix, enabling map alignment.

4.3.3 Fine-grained Refinement

One limitation of aligning the map using only a few key frames is that it does not adapt as users continue to encounter each other and exchange more features. Additionally, small orientation errors at the initial encounter point can accumulate and degrade accuracy when extrapolated over larger distances as users explore new areas. To address these issues, CooperSLAM incorporates a fine-grained refinement module that dynamically and progressively improves alignment.

CooperSLAM achieves this by performing local bundle adjustment to optimize key frames and map points within a sliding window in the local mapping thread. This optimization uses the LM algorithm [64] to minimize the reprojection error across all feature

measurements with the following loss function:

$$J(\mathbf{x}_{c_i}, {}^{G}\mathbf{x}_{f_i}) := \sum_{k=1}^{K} \sum_{j \in \mathcal{S}_{i,k}} \rho\left(\left\| {}^{C}\mathbf{z}_{i,j} - \mathbf{h}\left(\hat{\mathbf{x}}_{c_{i,k}}, {}^{G}\mathbf{x}_{f_{i,j}}\right) \right\|_{\Sigma}^{2} \right), \tag{4.6}$$

where $\mathcal{S}_{i,k}$ denotes a set of feature numbers that have matches at timestamp k, ρ denotes the Huber norm [46], $\mathbf{h}$ denotes the perspective projection function [105], ${}^{C}\mathbf{z}_{i,j}$ denotes the camera measurement for feature j from the camera frame of agent i, $\hat{\mathbf{x}}_{c_{i,k}}$ denotes the estimate of pose of camera frame at time step k, ${}^{G}\mathbf{x}_{f_{i,j}}$ is the registered position of feature j in the global frame of agent i, and Σ is the covariance matrix.

As users encounter each other, they also share records of past interactions. For example, if user A and B both encountered C before, their encounter logger would both have the records of C. If A and B meet each other, they can use information from C to help refine their alignment. Intuitively, this is because the records of C provide common ground that helps with the refinement. Specifically, CooperSLAM employs a global pose graph optimization (PGO) to correct the loop formed by A, B, and C, and refine the map alignment estimates. In this thread, we minimize $J(\mathbf{x}_{c_i}, {}^{G}\mathbf{x}_{f_i})$ for all $i = 1 : n$ in Eq. 4.6 on each agent, where we collected not only the local pose-feature constraints but also the common feature constraints from the received and reconstructed shadow key frames.

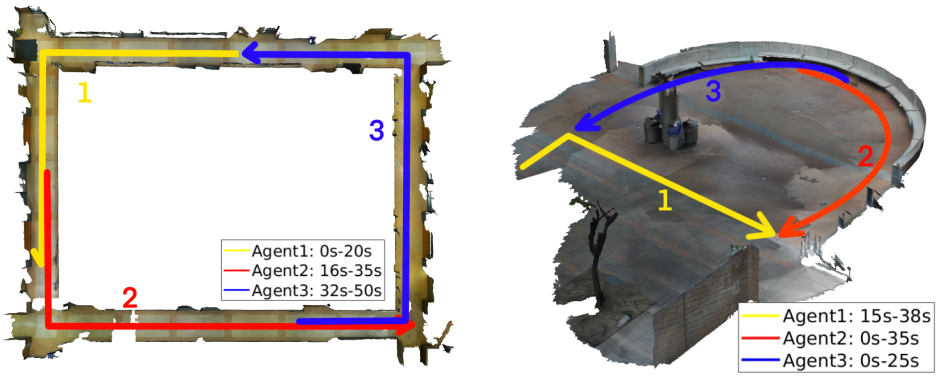
4.4 Experiments

In this section, we first evaluate the system in terms of localization accuracy, data transmission size, and map alignment latency, comparing it to existing approaches. Then we dive deeper into each component to analyze the sensitivity of each design choice and its

contribution. To evaluate the performance of CooperSLAM and compare its performance against various baselines, we use both real-world traces that we collect and publicly EuRoC [24] and Newer College [76] datasets.

4.4.1 Data Collection

We record our own dataset using three Zed2i cameras [6], capturing images from their stereo cameras along with IMU data. Figure 4.6 shows the environment of two scenarios used for collection: a circular hallway denoted as *Corridor*, and a half-circle courtyard denoted as *Yard*. The figure illustrates one example trace collected for each scene. In the corridor scenario, agent 1 (yellow) begins at the middle of the top hallway, turns left at the top-left corner, and encounters agent 2 (red) between approximately 16 and 20 seconds. agent 2 then proceeds toward the end of the bottom hallway, where it meets agent 3 for around 32 to 35 seconds. Notably, agent 3 does not encounter agent 1. The exercise concludes after 50 seconds. In the yard scenario, agents 2 and 3 start together, walking along the rim of the yard in opposite directions until they encounter agent 1 at approximately 15 seconds and 25 seconds, respectively, at the indicated locations. Unlike the corridor scenario, where each agent experiences only one encounter, the yard scenario involves two encounters per agent. To emulate these encounters, we recorded three clockwise and three counterclockwise traces for both the corridor and yard scenarios. Each trace includes more than two traversals.



(a) Corridor (b) Yard

Figure 4.6: Evaluation scenarios and traces. We collect traces in two scenarios, a rectangular hallway, denoted as corridor (a), and a half-circle courtyard, denoted as yard (b). The lines of different colors indicate trajectories from different users. The legend indicates the temporal interval of each spatial trajectory.

4.4.2 Baselines and Evaluation Metrics

We compare CooperSLAM against *state-of-the-art* C-SLAM baselines, as summarized in Table 4.1. The table highlights the properties of these baselines and showcases where CooperSLAM sets itself apart. Specifically, CCM-SLAM [81] shared entire key frames, including all associated data structures, via a centralized server. CarMap [15] assumes that each agent maintains a local copy of the global map and detects and shares only the differences of a map segment with the cloud. SwarmMap, built on top of CarMap, shares update instructions rather than raw update data with the server. All the above methods rely on a stable edge/cloud connection, making them vulnerable to packet losses. While SwarmMap can tolerate the loss of trivial update commands (e.g., incrementing the number of observations for a map point), losing more critical updates compromises relocalization accuracy. Additionally, CCM-SLAM and SwarmMap use monocular cameras instead of stereo cameras. To address this limitation and ensure a fair comparison, we employ $sim(3)$ alignment

Table 4.1: Feature comparison between baselines and the proposed algorithm w.r.t. four key aspects: infrastructure independence, map-sharing strategy, tolerance to packet loss, and collaboration method.

	Infrastructure-less	Map Sharing	Tolerate Packet Loss	C-SLAM Method
CCM-SLAM [81]	✗	Key Frames	✗	Centralized
CarMap [15]	✗	Segments	✗	Centralized
SwarmMap [95]	✗	Key Frames	✓	Centralized
CooperSLAM [Ours]	✓	Map Points	✓	Distributed

in OpenVINS [39], which compensates for the scale ambiguity in monocular estimates.

We evaluate CooperSLAM in terms of the following rules:

- **Localization Accuracy.** To compare the localization accuracy between the estimated trajectories and the ground truth, we use the absolute trajectory error (ATE) to evaluate both location and orientation accuracy. ATE indicates the average pose error over all frames. The precise definition of ATE is as follows:

$$e_{ATE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{x}_k \boxminus \hat{\mathbf{x}}_k^{+}\right\|_2^2} \tag{4.7}$$

  where, $\mathbf{x}_k$ is the ground truth pose, $\hat{\mathbf{x}}_k^{+}$ is the aligned estimated trajectory pose, and K denotes the number of poses/key frames. Note that $\boxminus$ is used because state vector involves quaternion.

- **Distance to Encounter (D_E).** In multi-user AR settings, we care about mapping accuracy at different locations in the environment, such as near where users encounter each other. Therefore we evaluate the ATE of all poses within the vicinity of the first relocalized inlier key frame, referred to as distance to encounter (DTE).

- **Data Transmission Size.** In totally decentralized settings, we care about data

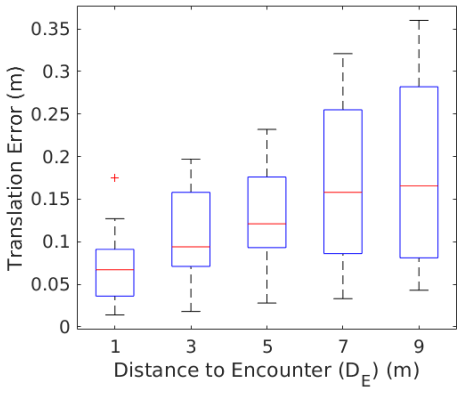


Figure 4.7: ATE within Different D_E.

transmission size as it directly impacts the alignment latency and battery life. For each baseline, we evaluate the total size of the data needed to be shared before getting good quality alignment as reflected by the ATE.

4.4.3 Key Results

We run CooperSLAM and all baselines in corridor, yard, EuRoC, and Newer College datasets. We collect 20 user encounters across the corridor and yard and run Monte Carlo simulations 10 times with random seeds for each encounter. Table 4.2 summarizes the key results across all the three dimensions we evaluate. In the table, we show the average

Table 4.2: Key Results Summary. Compared to baselines, CooperSLAM achieves smaller ATE within the interaction area, while requiring much less transmission data, and achieving map alignment faster.

	ATE ($D_E < 10$m)	Comm. Size / Frame (KB)	Est. Comm. Size to Alignment (KB)	Alignment Latency (s)
CCM-SLAM [81]	0.154±0.164	55±2	1396	4.9±1.0
CarMap [15]	0.216±0.101	47±8	829	2.1±0.5
SwarmMap [95]	0.204±0.061	60±5	375	1.5±0.2
CooperSLAM	**0.134±0.041**	**13±1**	**106**	**1.4±0.3**

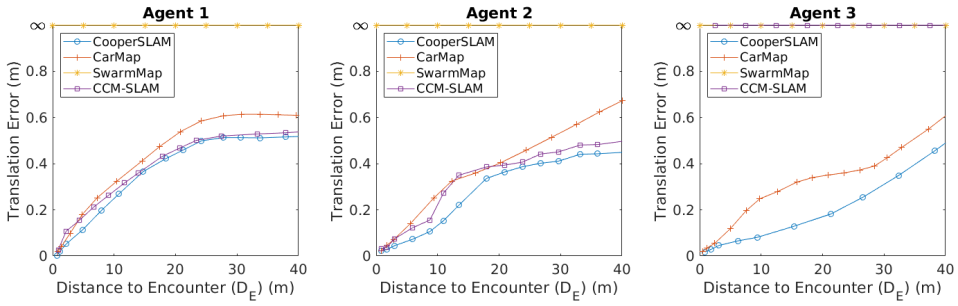


Figure 4.8: ATE comparison between CooperSLAM and baselines using three agents' trace from corridor

ATE within a distance to encounter of 5 m. Compared to all baselines, CooperSLAM performs the best with minimum trajectory errors. Figure 4.7 shows the ATE with different distances to encounter D_E from 0-10m. The average translation error is less than 10 cm within 2 meters, and below 15 cm within 10 meters.

For poses farther away from the encounter, we show the ATE of all three agents in one example trace for corridor dataset in Figure 4.8, and Vicon Room 1 from EuRoC in Figure 4.9. Different agents' trajectories are different, yielding different localization errors. CooperSLAM's improvement against other baselines also varies. In most of the cases within a small distance D_E, CooperSLAM slightly outperforms baselines. However, in the most extreme case, CooperSLAM outperforms CarMap by 66% (e.g. Agent 3 in corridor). In some of the other extreme cases, baseline methods even fail to align agents with each other (e.g. CCM-SLAM for Agent 3 and SwarmMap for Agent 2 in EuRoC traces).

In Figure 4.10, we illustrate the step-by-step process of Local SLAM, map switching, and global pose graph optimization using the Newer College dataset. Initially, each of the three agents operates local SLAM independently. When Agent 1 approaches the starting

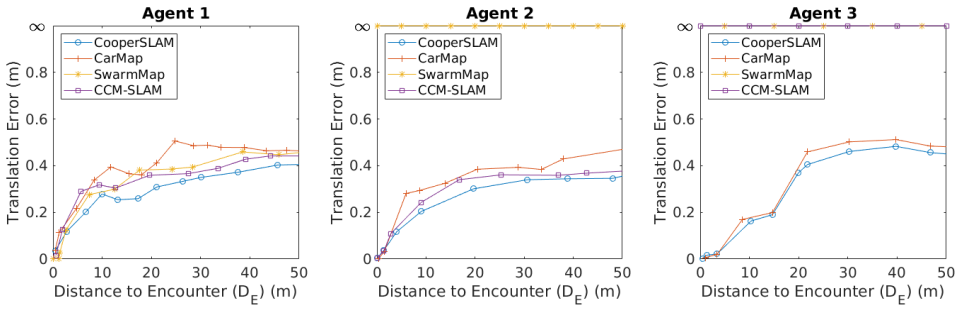


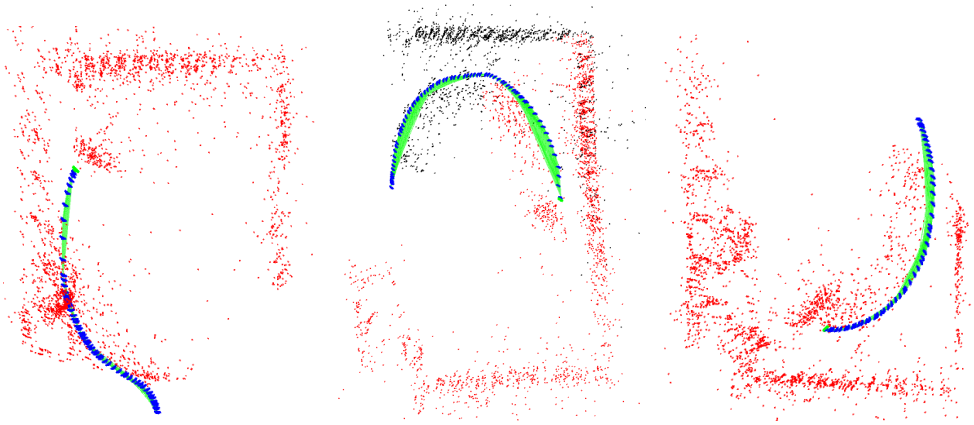
Figure 4.9: ATE comparison between CooperSLAM and baselines using three agents' trace from Vicon Room 1 of EuRoC

Table 4.3: The ATE of the orientation / position (degrees / meters) of three agents using Newer College sequences.

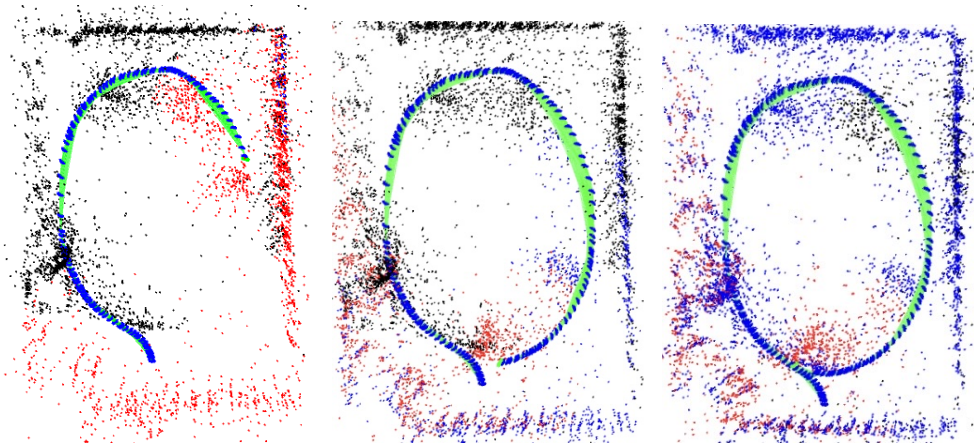
	CooperSLAM (wo PGO)	CooperSLAM
Agent 1	2.351 / 1.153	**1.800 / 0.946**
Agent 2	3.328 / 1.349	**2.356 / 1.056**
Agent 3	2.287 / 1.708	**1.261 / 0.994**

position of Agent 2, they exchange data and share their maps. Similarly, Agent 3 integrates all previously collected map points and keyframes during its data exchange. Finally, loop closure detection and pose graph optimization are performed to refine all keyframe poses and update the global map in a consistent manner. And, the ATE of localization accuracy before and after the pose graph refinement (last step in Figure 4.10) can be shown in Table 4.3. The results indicate that the global refinement can improve the localization accuracy by maintaining a consistence map.

Importantly, while achieving lower absolute trajectory error, CooperSLAM also requires much less data to be transmitted for the map alignment. Because different methods send various data structures for map alignment, to be fair, we compare both per frame data sizes (Table 4.2, 4th column) under different schemes, as well as total data size to achieve



(a) Agent 1 (b) Agent 2 (c) Agent 3



(d) Stitch 1 & 2 (e) Stitch 1 & 2 & 3 (f) PGO

Figure 4.10: Step-by-step process of Local SLAM, map switching, and global pose graph optimization using the Newer College dataset.

alignment (Table 4.2, 5th column). CooperSLAM's flexible map point sharing requires an average of only 13KB for each frame and 106KB to achieve alignment, whereas all baselines need over 40KB per frame and up to 13x more data to achieve the alignment of quality shown in Table 4.2.

In addition to sending less data, CooperSLAM is also robust to packet loss and does not trigger retransmissions to delay the initial alignment. The last column of Table 4.2

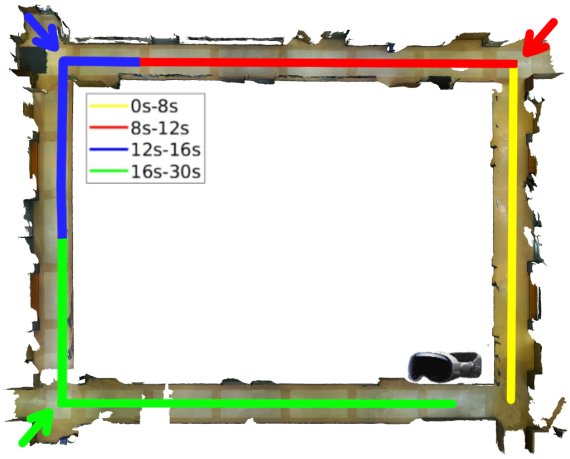


Figure 4.11: Loss Rate Measurement Trajectory

shows that CooperSLAM needs only 1.4 s to finish the alignment, which is the lowest among all baselines. Figure 4.12 shows a latency profile of sampled map point transmissions. On average, transmissions takes less than 3.0 ms, while saving and loading together take less than 1.7 ms.

4.4.4 Sensitivity Analysis

In this subsection, we dive deeper into each component. We study the localization robustness to packet loss, evaluate the sensitivity of alignment consensus to the number of shadow key frames available, validate the outlier rejection algorithm, and understand map refinement performance.

- **Packet Loss.** To evaluate how much packet loss CooperSLAM can tolerate and understand where does the system break, we first measure the packet loss rate in a real-world setting (corridor) and then play back the traces while injecting the same

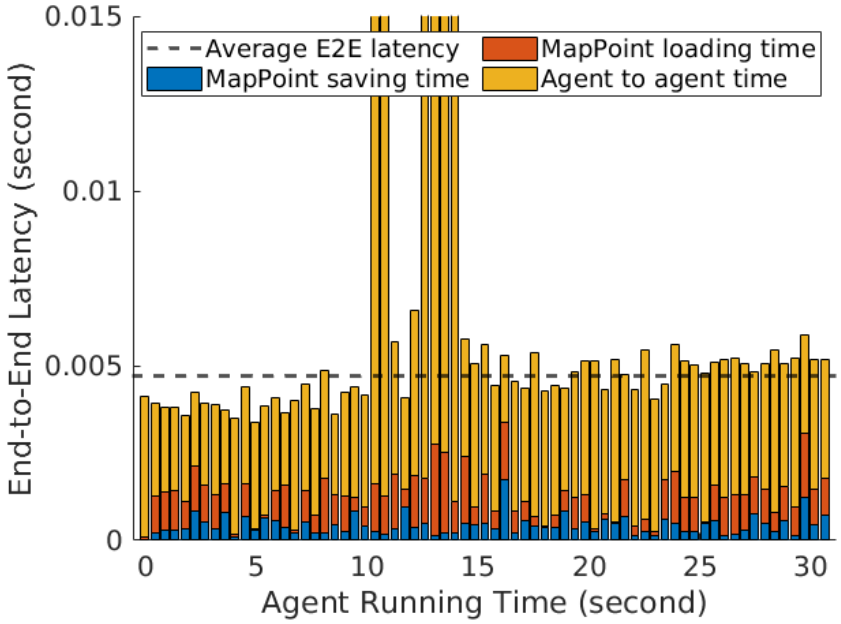


Figure 4.12: Map Point Latency Profile

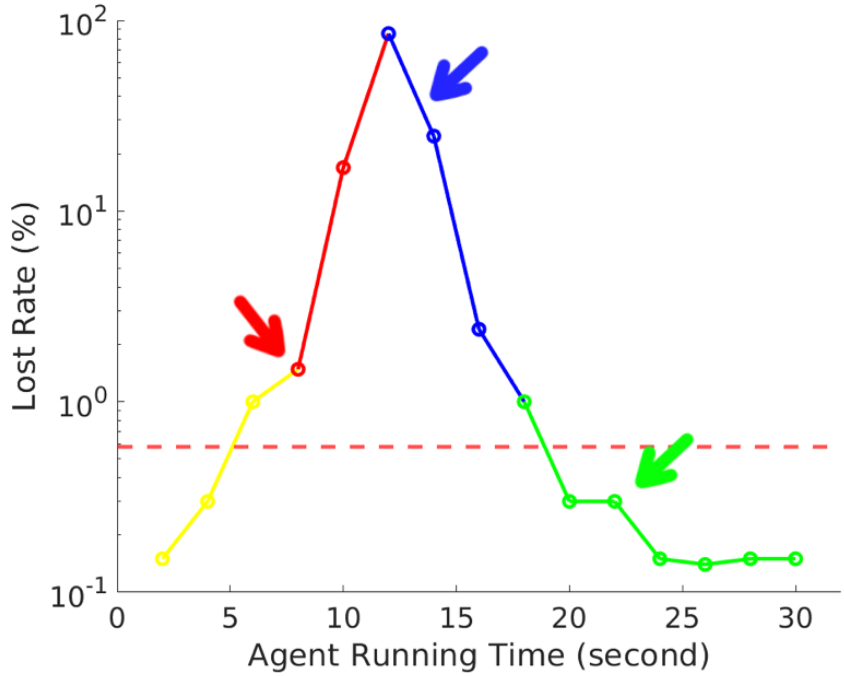
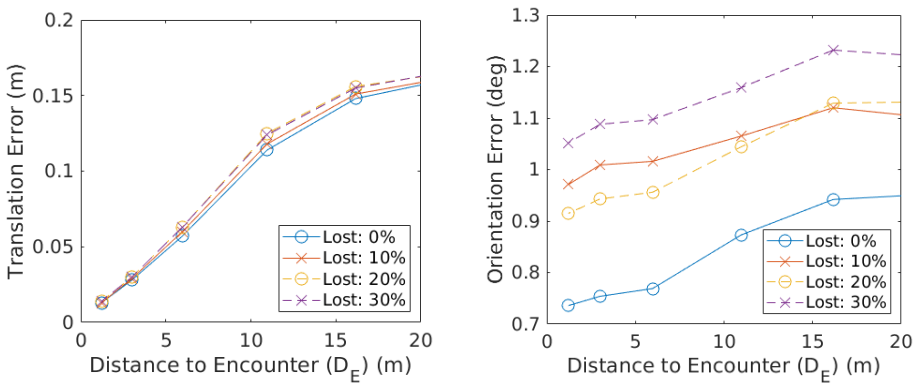


Figure 4.13: Packet Lose Rate over Time



(a) Translation error

(b) Orientation error

Figure 4.14: ATE under Different Map Points Loss Rates

packet loss rate to evaluate the localization accuracy.

First, to measure the packet loss rate, we put two agents in the corridor scene as shown in Figure 4.11; one stationary agent located at the right bottom corner (marked as a Vision Pro headset), who is connected to a WiFi router using an Ethernet cable. The other agent (connected to the router wirelessly) walks the full loop of the hallway starting from the bottom right corner. Each color represents a temporal segment; the agent starts from second 0, arrives at the end of the right hallway at time 8 s (yellow), turns left and walks through the top hallway in about 4 seconds (red), then makes its way around the corner towards the middle of the left hallway (blue) and finishes the rest at around 30s (green).

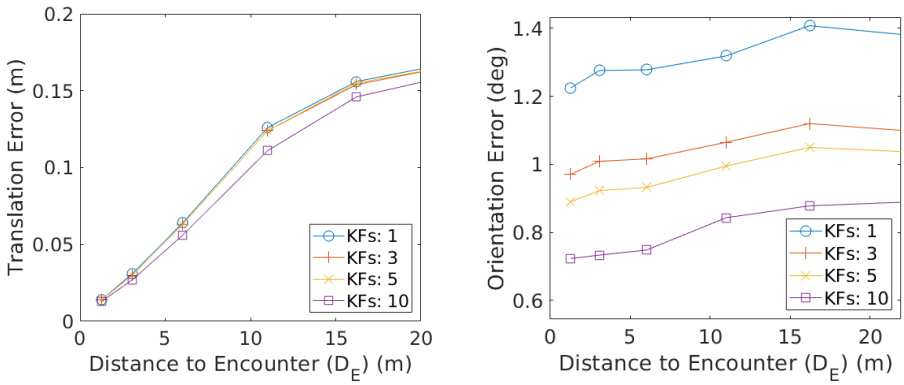
We measure the packet loss rate at 2 second intervals, as shown in Figure 4.13. The maximum loss rate occurs when the agent is in the top hallway (red) and in the farthest diagonal corner (blue) from the stationary agent. When the loss rate hits the peak in Figure 4.13, the link is basically disconnected with 100% packet loss (no

real-time AR visualizations or interactions possible). When the second agent traverses the green and yellow sections, the loss rate is less than 2%. The loss rate consistently correlates with the latency numbers in Figure 4.12.

Now to understand when the system breaks, we play back the corridor traces and inject uniform packet losses at various rates. Figure 4.14 shows the translation error and orientation error when up to 30% of map points are lost. The impact of missing map points is almost negligible; the difference in translation error within $D_E < 20$ m is under 1 cm, and the difference in orientation error is under 0.3 degrees. The only visible difference on translation is when the distance to encounter $D_E > 40$ m; the translation error difference is caused by the extrapolation based on slightly erroneous orientation (less than 1 degree off) at the matched frames ($D_E = 0$). Note also that because the extrapolation is based on a few key frames' alignment, the trend in orientation error under high or low loss rate is consistent.

Since map alignment is a premise for precise 3D AR interactions, CooperSLAM should tolerate a higher loss rate than what is sustainable to support real-time interactions. Figure 4.14 shows that CooperSLAM's map alignment can tolerate very high packet loss rates. Nevertheless, we also conducted a stress test. We find that CooperSLAM completely fails if the missing map points exceed 60%.

- **Key frame Synthesis.** Next, we evaluate the map alignment performance using different numbers of shadow key frames to understand its sensitivity. In this evaluation, we assume no packet loss. Specifically, during an encounter, we let CooperSLAM transmit and collect map points until sufficient shadow key frames can be



(a) Translation error

(b) Orientation error

Figure 4.15: ATE using Different Numbers of Shadow Key frames

reconstructed and accurately relocalized. Figure 4.15 shows the ATE over D_E using $KFs \in [1, 3, 5, 10]$ key frames. Similar to Figure 4.14, the ATE is not very sensitive to increasing number of shadow key frames being used, especially at small D_E, e.g. when close to the encounter position. When $D_E < 5m$, synthesizing ten key frames rather than one improves translation error by < 1 cm and orientation error by < 0.2 degree. The extrapolation from the orientation again factors in at longer distances where the improvements with more shadow key frames can be up to 5 cm.

- **Outlier Rejection.** To validate CooperSLAM's alignment outlier rejection policy, we collect all of the relocalized frames from all encounters in corridor and yard and randomly select 100 samples to study the statistics of the relocalization error distribution. Figure 4.16 shows the distribution of translation (x, y, z) and orientation (yaw, pitch, roll). The red dashed lines denote the left and right boundaries of 99% confidential intervals. The results indicate that, for a high quality key frame relocalization, errors above 0.5m in translation, 0.5 rad in yaw and roll, or 0.02 rad in pitch,

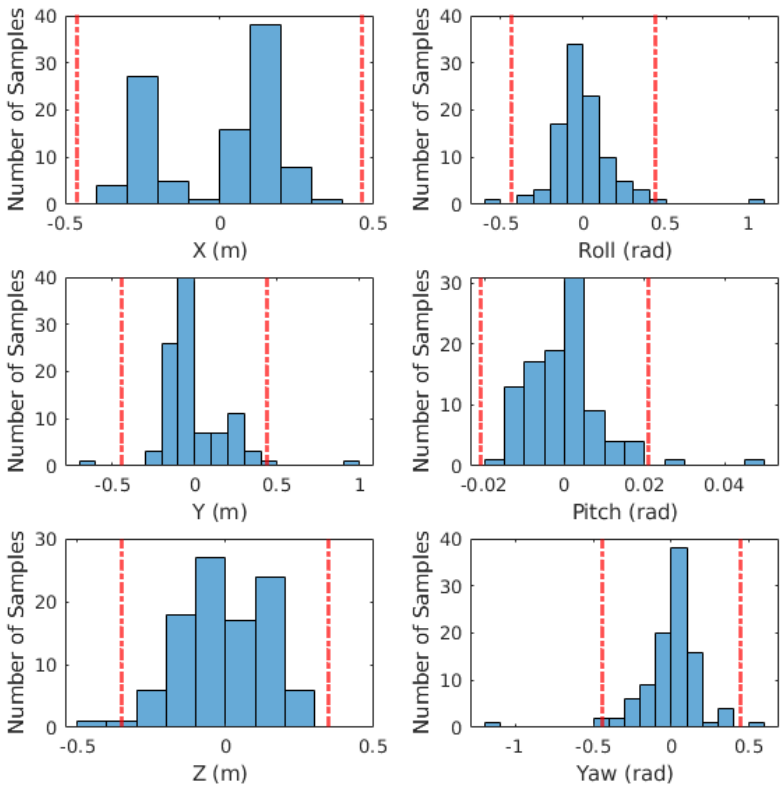
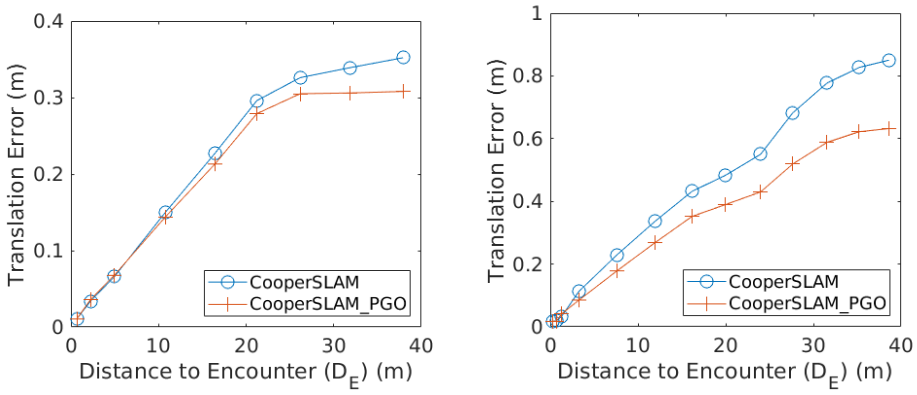


Figure 4.16: Transformation matrix error distribution (shown in x, y, z, roll, pitch, yaw). The red dashed lines indicate the 99% confidence intervals.

are relatively rare. It is interesting to see the distribution for pitch and z is more concentrated than other dimension. This is possibly an artifact of our rigid sensor setup and our horizontal motion pattern, such that there is not much vertical motion or rotation around the pitch axis. Nevertheless, the results suggest that sampling a 99% confidence interval within local records of relocalized frames would provide sufficient statistics to reject outliers.

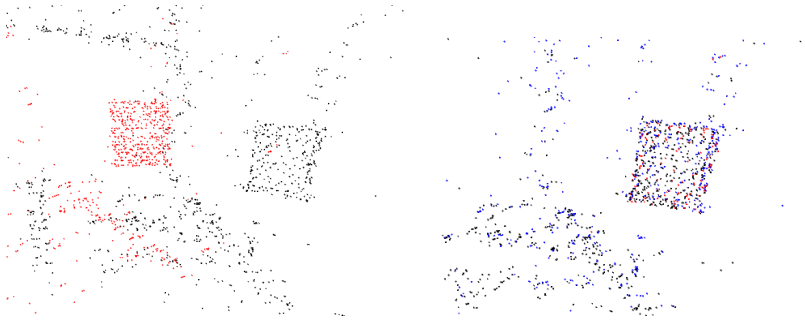
- **Map Refinement.** Finally, we evaluate the contribution of CooperSLAM's map



(a) Corridor (b) Yard

Figure 4.17: Comparison of ATE with and without Map Refinement ("CooperSLAM_PGO" and "CooperSLAM", respectively).

refinement module. We compare the ATE with and without the refinement process in both corridor and yard encounters. In this evaluation, we also assume no packet loss. As shown in Figure 4.17, map refinement improves ATE by up to approximately 16% in the corridor scenario and up to about 30% in the yard scenario. The interesting observation is that while the improvement may not look significant when $D_E < 10$m, because of the extrapolation of position based on orientation, a small improvement in orientation can go a long way in correcting the translation error where the distance to encounter is from 10 to 40 meters. This is helpful when some virtual objects left behind by one user are revisited by other agents, or when agent encounters form a loop (e.g. the yard trace shown in Figure 4.6 (b)). After exchanging encounter logs, commonly encountered agent history can be exchanged to provide a good initial guess of the transformation matrix, since the two devices participating in the current encounter can use the coordinate system of the previously encountered common agent. These



(a) One-shot relocalization (b) Refined relocalization

Figure 4.18: The comparison between single key frame relocalization without refinement (CarMap) and with a local pose graph optimization refinement (CooperSLAM) using the EuRoC dataset.

records help refine the alignment because they can add common feature constraints to the global pose graph optimization as shown in Figure 4.18. We can figure out that after applying pose graph optimization, the feature points of the checkerboard align perfectly.

4.5 Conclusions

In this chapter, we explore the problem of enabling rapid on-the-fly AR interactions using infrastructure-less ad hoc cooperative SLAM. We design and implement CooperSLAM, which exchanges lightweight map points among AR users over intermittent P2P connections to achieve high-quality map alignment. Evaluations compared to *state-of-the-art* baselines show that while transmitting 71%-92% less data, CooperSLAM is still able to achieve 13%-38% better localization accuracy. Benefiting from the lightweight tiny messaging design, CooperSLAM is robust to packet losses, requires no infrastructure support, and enables interactions with minimal map synchronization delay. We believe that CooperSLAM fills an important gap in enabling infrastructure-less mobile AR immersive experiences.

Chapter 5

Conclusions

This dissertation has presented a series of innovative methodologies to address the pressing challenges in multi-user AR systems, focusing on advancing VIO and C-SLAM techniques. The contributions span multiple aspects of AR localization, mapping, and multi-user collaboration, with detailed experimental validation across simulation and real-world datasets.

The PL-CVIO framework addresses the limitations of traditional VIO methods in low-feature environments by integrating both point and line features. Through a novel cooperative approach, neighboring robots share observations of common features, enabling enhanced localization accuracy and robustness. The use of CI ensures consistent state estimation while maintaining computational efficiency, which is critical for multi-robot scenarios. Experimental results demonstrate that the proposed PL-CVIO framework achieves significant improvements over existing VIO and cooperative VIO methods, particularly in environments with sparse features, such as human-made indoor spaces.

The integration of NeRF into map-based VIO represents a transformative approach to leveraging AI-driven models for localization and mapping. By encoding compact, photorealistic representations of 3D environments, NeRF provides a powerful tool for addressing challenges related to initialization, drift correction, and robustness in dynamic environments. A novel pose initialization model using geodesic errors on $SE(3)$ and an online NeRF-based VIO algorithm were developed to fully exploit the NeRF prior map. The experimental results validate that NeRF-based VIO improves both pose accuracy and robustness to environmental alterations, making it a promising direction for real-world AR applications.

The CooperSLAM algorithm tackles the challenges of infrastructure dependency in multi-user AR systems. By utilizing lightweight peer-to-peer communication and sparse map feature sharing, CooperSLAM enables collaborative mapping and localization in dynamic, infrastructure-less settings. This approach decouples map points and key frames, leveraging opportunistic encounters between users to align maps efficiently. The algorithm demonstrated scalability and robust performance in diverse scenarios, including disaster recovery and remote exploration, where centralized infrastructure is unavailable.

The combined contributions of this dissertation establish a robust foundation for the next generation of AI-aided multi-user AR systems. By addressing the challenges of feature sparsity, map initialization, and infrastructure dependence, this work significantly advances the *state-of-the-art* in localization and mapping for AR. The extensive evaluations using both synthetic and real-world datasets highlight the practical feasibility of the proposed solutions, confirming their potential for deployment in various applications, including

immersive gaming, collaborative robotics, and emergency response.

Future work can extend these methodologies by exploring additional sensing modalities, such as lidar or depth sensors, for environments with poor visual cues. Furthermore, integrating learning-based approaches to enhance feature extraction and tracking in complex or dynamic environments could further improve performance. Optimizing computational efficiency for deployment on lightweight and resource-constrained AR devices also represents a critical avenue for future research. Overall, this dissertation establishes a foundation for developing AR systems that are robust, scalable, and capable of functioning effectively in dynamic and resource-constrained environments. By addressing key challenges in multi-agent SLAM and data efficiency, this work contributes to the broader advancement of AR technologies, enabling their application across a wide range of domains.

Bibliography

[1] Apple vision pro. https://www.apple.com/apple-vision-pro.

[2] Google arcore. https://developers.google.com/ar.

[3] Intel realsense d455. https://www.intelrealsense.com.

[4] Meta quest 3. https://developer.oculus.com/meta-quest-3.

[5] Vicon. https://www.vicon.com.

[6] Zed2i camera. https://www.stereolabs.com/products/zed-2.

[7] Apple arkit. https://developer.apple.com/augmented-reality/arkit/, 2023.

[8] Bmw m mixed reality. https://www.bmw-m.com/en/topics/magazine-article-pool/m-mixed-reality.html, 2023.

[9] Catan: World explorers. https://en.wikipedia.org/wiki/Catan:_World_Explorers, 2023.

[10] Mercedes mbux. https://www.mercedes-benz.com.my/passengercars/mercedes-benz-cars/mbux/mbux-stage.module.html, 2023.

[11] New maps updates: Immersive view for routes and other ai features. https://blog.google/products/maps/google-maps-october-2023-update/, 2023.

[12] New ways maps is getting more immersive and sustainable. https://blog.google/products/maps/sustainable-immersive-maps-announcements/, 2023.

[13] Pokemon go. https://pokemongolive.com/, 2023.

[14] Wallame. https://en.wikipedia.org/wiki/WallaMe, 2023.

[15] Fawad Ahmad, Hang Qiu, Ray Eells, Fan Bai, and Ramesh Govindan. CarMap: Fast 3d feature map updates for automobiles. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 1063–1081, Santa Clara, CA, February 2020. USENIX Association.

[16] Kittipat Apicharttrisorn, Jiasi Chen, Vyas Sekar, Anthony Rowe, and Srikanth V. Krishnamurthy. Breaking edge shackles: Infrastructure-free collaborative mobile augmented reality. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 1–15, New York, NY, USA, 2023. Association for Computing Machinery.

[17] Alireza Bahremand, Linda Nguyen, Tanya Harrison, and Robert LiKamWa. Hololucination: A framework for live augmented reality presentations across mobile devices. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 243–2431. IEEE, 2019.

[18] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, 2005.

[19] Ali J. Ben Ali, Marziye Kouroshli, Sofiya Semenova, Zakieh Sadat Hashemifar, Steven Y. Ko, and Karthik Dantu. Edge-slam: Edge-assisted visual simultaneous localization and mapping. *ACM Trans. Embed. Comput. Syst.*, 22(1), oct 2022.

[20] Daniele Benedettelli, Andrea Garulli, and Antonio Giannitrapani. Cooperative slam using m-space representation of linear features. *Robotics and Autonomous Systems*, 60(10):1267–1278, 2012.

[21] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 298–304, 2015.

[22] Chiara Boretti, Philippe Bich, Yanyu Zhang, and John Baillieul. Visual navigation using sparse optical flow and time-to-transit. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9397–9403, 2022.

[23] G Bradski. The opencv library. dr. dobb's journal: Software tools for the professional programmer. 2000.

[24] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[25] J.C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

[26] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[27] M Jorge Cardoso, Tal Arbel, Gustavo Carneiro, Tanveer Syeda-Mahmood, João Manuel RS Tavares, Mehdi Moradi, Andrew Bradley, Hayit Greenspan, João Paulo Papa, Anant Madabhushi, et al. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings*, volume 10553. Springer, 2017.

[28] Julie Carmigniani, Borko Furht, Marco Anisetti, Paolo Ceravolo, Ernesto Damiani, and Misa Ivkovic. Augmented reality technologies, systems and applications. *Multimedia tools and applications*, 51:341–377, 2011.

[29] Chuchu Chen, Patrick Geneva, Yuxiang Peng, Woosik Lee, and Guoquan Huang. Monocular visual-inertial odometry with planar regularities. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6224–6231, 2023.

[30] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford. Deep learning features at scale for visual place recognition. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230, 2017.

[31] Titus Cieslewski, Siddharth Choudhary, and Davide Scaramuzza. Data-efficient decentralized visual slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2466–2473, 2018.

[32] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0 – a modular and multimodal mapping framework. *IEEE Robotics and Automation Letters*, 8(2):520–527, 2023.

[33] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018.

[34] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

[35] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981.

[36] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014.

[37] Jean Gallier. *Geometric methods and applications: for computer science and engineering*, volume 38. Springer Science & Business Media, 2011.

[38] Dorian Galvez-López and Juan D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[39] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. Openvins: A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020.

[40] Patrick Geneva and Guoquan Huang. Map-based visual-inertial localization: A numerical study. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7973–7979, 2022.

[41] Marcel Geppert, Peidong Liu, Zhaopeng Cui, Marc Pollefeys, and Torsten Sattler. Efficient 2d-3d matching for multi-camera visual localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5972–5978, 2019.

[42] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.

[43] Yijia He, Ji Zhao, Yue Guo, Wenhao He, and Kui Yuan. Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features. *Sensors*, 18(4), 2018.

[44] Wang Hu. *Optimization-Based Risk-Averse Outlier Accommodation With Linear Performance Constraints: Real-Time Computation and Constraint Feasibility in CAV State Estimation*. University of California, Riverside, 2024.

[45] Guoquan P. Huang, Nikolas Trawny, Anastasios I. Mourikis, and Stergios I. Roumeliotis. On the consistency of multi-robot cooperative localization. In *Robotics: Science and Systems V*. The MIT Press, 07 2010.

[46] Peter J. Huber. *Robust Estimation of a Location Parameter*, pages 492–518. Springer New York, New York, NY, 1992.

[47] Hangil Kang, Hoyoung Kim, and Young Min Kwon. Recen:resilient manet based centralized multi robot system using mobile agent system. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1952–1958, 2019.

[48] Nadir Karam, Frederic Chausse, Romuald Aufrere, and Roland Chapuis. Localization of a group of communicating vehicles by state exchange. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 519–524, 2006.

[49] Anton Kasyanov, Francis Engelmann, Jörg Stückler, and Bastian Leibe. Keyframe-based visual-inertial online slam with relocalization. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6662–6669, 2017.

[50] Saimouli Katragadda, Woosik Lee, Yuxiang Peng, Patrick Geneva, Chuchu Chen, Chao Guo, Mingyang Li, and Guoquan Huang. Nerf-vins: A real-time neural radiance field map-based visual-inertial navigation system, 2023.

[51] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[52] Pierre-Yves Lajoie, Benjamin Ramtoula, Yun Chang, Luca Carlone, and Giovanni Beltrame. Door-slam: Distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020.

[53] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[54] Adrian S Lewis and Michael L Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141:135–163, 2013.

[55] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[56] Mingyang Li and Anastasios I. Mourikis. Online temporal calibration for camera–imu systems: Theory and algorithms. *The International Journal of Robotics Research*, 33(7):947–964, 2014.

[57] Chun Liu and Andreas Kroll. A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. In Leszek Rutkowski, Marcin Korytkowski, Rafał Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing*, pages 466–474, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[58] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[59] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[60] Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, Vancouver, Canada, August 1981.

[61] Ryan Luna and Kostas E. Bekris. Efficient and complete centralized multi-robot path planning. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3268–3275, 2011.

[62] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-nerf: Monte carlo localization using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4018–4025, 2023.

[63] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[64] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[65] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.

[66] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[67] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[68] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[69] Esha D. Nerurkar, Stergios I. Roumeliotis, and Agostino Martinelli. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *2009 IEEE International Conference on Robotics and Automation*, pages 1402–1409, 2009.

[70] Esha D. Nerurkar, Kejian J. Wu, and Stergios I. Roumeliotis. C-klam: Constrained keyframe-based localization and mapping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3638–3643, 2014.

[71] Mrinal K. Paul, Kejian Wu, Joel A. Hesch, Esha D. Nerurkar, and Stergios I. Roumeliotis. A comparative analysis of tightly-coupled monocular, binocular, and stereo vins. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 165–172, 2017.

[72] Peter Petersen. *Riemannian geometry*, volume 171. Springer, 2006.

[73] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508, 2017.

[74] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[75] Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3662–3669, 2018.

[76] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360, 2020.

[77] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, 2016.

[78] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

[79] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[80] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[81] Patrik Schmuck and Margarita Chli. Ccm-slam: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, 36(4):763–781, 2019.

[82] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. Maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018.

[83] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687, 2018.

[84] Sayem Mohammad Siam and Hong Zhang. Fast-seqslam: A fast appearance based place recognition algorithm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5702–5708, 2017.

[85] Felipe O. Silva, Álvaro H. A. Maia, Jean-Bernard Uwineza, Farzana S. Rahman, Zeyi Jiang, Wang Hu, and Jay A. Farrell. Dual-antenna gnss-aided ins stationary alignment with sensor parameter estimation. *IEEE Transactions on Instrumentation and Measurement*, 74:1–16, 2025.

[86] James M Sloan, Keith A Goatman, and J Paul Siebert. Learning rigid image registration-utilizing convolutional neural networks for medical image registration. 2018.

[87] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[88] Nikolas Trawny and Stergios I Roumeliotis. Indirect kalman filter for 3d attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2:2005, 2005.

[89] Ana Villanueva, Zhengzhe Zhu, Ziyi Liu, Kylie Peppler, Thomas Redick, and Karthik Ramani. Meta-ar-app: An authoring platform for collaborative augmented reality in stem classrooms. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[90] Deepak Geetha Viswanathan. Features from accelerated segment test (fast). In *Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK*, pages 6–8, 2009.

[91] Junyi Wang and Yue Qi. A multi-user collaborative ar system for industrial applications. *Sensors*, 22(4), 2022.

[92] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[93] Kejian J. Wu, Ahmed M. Ahmed, Georgios A. Georgiou, and Stergios I. Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems*. 07 2015.

[94] Jie Xu, Pengxiang Zhu, Yanyu Zhang, and Wei Ren. Moving target estimation and active tracking in multi-robot systems. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 6972–6977, 2023.

[95] Jingao Xu, Hao Cao, Zheng Yang, Longfei Shangguan, Jialin Zhang, Xiaowu He, and Yunhao Liu. SwarmMap: Scaling up real-time collaborative visual SLAM at the

edge. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 977–993. USENIX Association, 2022.

[96] Yulin Yang, Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Visual-inertial odometry with point and line features. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2447–2454, 2019.

[97] Yulin Yang and Guoquan Huang. Aided inertial navigation: Unified feature representations and observability analysis. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3528–3534, 2019.

[98] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330, 2021.

[99] Hongsheng Yu and Anastasios I. Mourikis. Vision-aided inertial navigation with line features and a rolling-shutter camera. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 892–899, 2015.

[100] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[101] Fuzhen Zhang. Quaternions and matrices of quaternions. *Linear Algebra and its Applications*, 251:21–57, 1997.

[102] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013.

[103] Yanyu Zhang, Marcus Greiff, Wei Ren, and Karl Berntorp. Distributed road-map monitoring using onboard sensors. In *2024 American Control Conference (ACC)*, pages 5049–5054, 2024.

[104] Yanyu Zhang, Xiu Wang, Xuan Wu, Wenjing Zhang, Meiqian Jiang, and Mahmood Al-Khassaweneh. Intelligent hotel ros-based service robot. In *2019 IEEE International Conference on Electro Information Technology (EIT)*, pages 399–403, 2019.

[105] Yanyu Zhang, Pengxiang Zhu, and Wei Ren. Pl-cvio: Point-line cooperative visual-inertial odometry. In *2023 IEEE Conference on Control Technology and Applications (CCTA)*, pages 859–865, 2023.

[106] Zhuqing Zhang, Yanmei Jiao, Shoudong Huang, Rong Xiong, and Yue Wang. Map-based visual-inertial localization: Consistency and complexity. *IEEE Robotics and Automation Letters*, 8(3):1407–1414, 2023.

[107] Lipu Zhou, Shengze Wang, and Michael Kaess. Dplvo: Direct point-line monocular visual odometry. *IEEE Robotics and Automation Letters*, 6(4):7113–7120, 2021.

[108] Pengxiang Zhu, Yulin Yang, Wei Ren, and Guoquan Huang. Cooperative visual-inertial odometry. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13135–13141, 2021.

[109] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam, 2023.

[110] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12786–12796, June 2022.

[111] Xingxing Zuo, Nathaniel Merrill, Wei Li, Yong Liu, Marc Pollefeys, and Guoquan Huang. Codevio: Visual-inertial odometry with learned optimizable dense depth. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14382–14388, 2021.

[112] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual slam with point and line features. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1775–1782, 2017.