CornerVINS: Accurate Localization and Layout Mapping for Structural Environments Leveraging Hierarchical Geometric Representations

Yidi Zhang, Fulin Tang, and Yihong Wu

***Abstract*—A compact and consistent map of surroundings is critical for intelligent robots to understand their situations and realize robust navigation. Most existing techniques rely on infinite planes, which are sensitive to pose drift and may lead to confusing maps. Toward high-level perception in indoor environments, we propose CornerVINS, an innovative RGB-D inertial localization and layout mapping method leveraging hierarchical geometric features, i.e., points, planes, and box corners. Specifically, points are enhanced by fusing depth information, and planes are modeled as bounded patches using convex hulls to increase their discriminability. More importantly, box corners, lying at the intersection of three orthogonal planes, are parameterized with a 6-D vector and integrated into the extended Kalman filter for the first time. We introduce a hierarchical mechanism to effectively extract and associate planes and corners, which are considered as layout components of scenes and serve as long-term landmarks to correct camera poses. Extensive experiments prove that the proposed box corners bring significant improvements, enabling accurate localization and consistent layout mapping at low computational cost. Overall, the proposed CornerVINS outperforms state-of-the-art systems in both accuracy and efficiency.**

***Index Terms*—Hierarchical geometric features, layout mapping, simultaneous localization and mapping (SLAM), visual-inertial navigation system.**

I. Introduction

VISUAL inertial odometry (VIO) and simultaneous localization and mapping (VI-SLAM), aiming to accurately estimate sensor poses and scene structures, have been widely studied for several decades [1], [2], [3], [4], [5]. These technologies have numerous applications, such as unmanned aerial vehicles [6], service robots [7], and augmented/virtual reality [8], [9].

Received 21 January 2025; accepted 13 April 2025. Date of publication 6 May 2025; date of current version 30 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62202468 and in part by the Beijing Science and Technology Plan Project Z231100007123005. This article was recommended for publication by Associate Editor G. Huang and Editor J. Civera upon evaluation of the reviewers' comments. *(Corresponding authors: Fulin Tang; Yihong Wu.)*

The authors are with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100190, China, and also with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: fulin.tang@nlpr.ia.ac.cn; yhwu@nlpr.ia.ac.cn).

The core implementations are available at https://github.com/zydddd/CornerVINS.

Digital Object Identifier 10.1109/TRO.2025.3567532

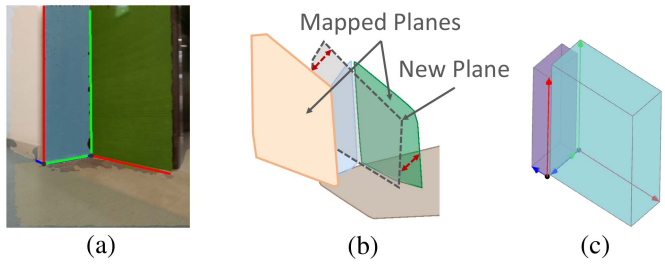

Fig. 1. (a) Example of planar ambiguous regions. The planes are assigned different colors and the corners with three axes are marked. (b) Similar planes can not be associated reliably with landmarks in the global map. (c) Box corners help with understanding the scene at a high level and distinguishing the scene effectively.

Complementary to point features, high-level features, such as planes, which exist prevalently in man-made environments and encode geometrical information, are more interpretable and usable in providing a structural representation of surroundings, and thus, can be exploited to further constrain motions, especially when points degenerate in challenging scenes [10], [11]. Besides, with the help of depth sensors, the accuracy and robustness of visual-inertial navigation systems (VINSs) are remarkably improved, as 3-D features can be directly extracted and associated with mapped landmarks, effectively limiting drift by reducing dependencies on successive frames [12]. Combining point and plane features using RGB-D cameras has demonstrated strong performance in indoor navigation [13], [14].

However, planes are not always distinguishable enough for localization purposes in practice [15], [16]. Although they can provide long-term constraints for soft relocalization, the plane-based motion estimation would be problematic when data association fails in ambiguous regions, as exemplified in Fig. 1. An acceptable way to improve the distinguishability of planes is modeling them as finite regions, which not only enhances the accuracy of data association but also produces a more realistic map [17], [18]. Nevertheless, their performance degrades in the presence of clutter. Some methods jointly utilize a group of planes to obtain robust associations [19], [20]. Since more information is encoded by combining multiple features, the chances of similar situations decrease, facilitating the system to recognize the scene and localize accurately. Moreover, high-level geometric features that integrate structural, semantic, and topological constraints between planes show promising values

in intelligent scene exploration, paving the way for localization and mapping of indoor service robots [20].

To make full use of plane features and high-level geometric information, we focus on a kind of iconic region in structured indoor environments, namely box corners, to support robust data association and provide long-term constraints for accurate localization and layout mapping. A box corner, as depicted in Fig. 1, consists of three mutually orthogonal planes and naturally forms a local coordinate system with their intersection as the origin and their normals as three axes. It has six degrees of freedom (DoF), including a 3-DoF rotation and a 3-DoF position, providing full constraints for robot motions. To the best of our knowledge, we are the first to utilize box corners in a visual-inertial system. On one hand, they usually appear in stable and static areas of indoor environments, such as walls, tables, and cabinets, and impose perpendicular constraints among planes. On the other hand, they can be robustly and efficiently extracted and associated based on the presence of planes with little ambiguity. The alignment of box corners also effectively assists in the association of planes, improving the positioning accuracy and consistency of the generated map. The box corners build a bridge between geometric and semantic features, offering the chance to understand the layout of surrounding environments at a high level. These characteristics make box corners highly informative to be utilized in improving localization accuracy and constructing consistent and interpretable maps for VINSs in structured indoor scenes.

Therefore, in this article, we fully exploit the potential of box corners and combine them with point and plane features in the extended Kalman filter (EKF) framework using an RGB-D camera and an inertial measurement unit (IMU). The main contributions of this work are summarized as follows.

1) A stable and distinctive feature in man-made buildings, box corner, which seamlessly integrates the structural regularity of planes, is first proposed and parameterized as a 6-D vector.
2) A hierarchical mechanism is adopted for plane and box corner features, to support accurate, efficient, and robust feature extraction, representation, and association.
3) An innovative RGB-D inertial navigation system leveraging hierarchical geometric features, called CornerVINS, is developed for indoor localization and layout mapping. It fully utilizes depth information and explores layout-level constraints to correct accumulated errors.
4) Extensive experiments demonstrate that CornerVINS achieves high localization accuracy and map consistency with low computational cost, validating the effectiveness of the proposed novelties in real-world circumstances.

II. Related Work

In recent years, depth sensors have gained popularity for their abilities to provide 3-D information that can be utilized in VINSs to enhance the point triangulation quality or construct residuals [4], [12], [18], [21], [22], [23], [24]. In artificial environments, planes contain high-level geometrical information, which can be extracted directly from an RGB-D camera and serve as extra measurements complementary to points to further constrain motions [13], [25], [26], [27], [28]. Some studies enforce point-to-plane and homography constraints to improve the localization accuracy [13], [18], [24], [29], [30].

Moreover, geometric constraints between planes, such as parallelism and orthogonality, have been proven to be effective for improving camera pose estimation accuracy in structured indoor environments [27]. Some methods model surrounding environments with known assumptions to reduce drift and simplify feature representation [9], [31], [32]. A strong assumption is Manhattan world (MW) [31], [33], where only three mutually perpendicular heading directions are defined. Another common assumption is Atlanta world (AW) [32], [34], which contains multiple MWs sharing the same perpendicular orientation. Furthermore, Hong Kong world [35] and the mixture of MW [36] support more general scene structures. By aligning the main directions of these world assumptions, the drift-free rotation can be estimated and the translation is computed in bundle adjustment (BA) with fixed rotation [37], [38]. In [39] and [35], main directions also participate in the optimization process.

Regardless of the assumptions, data association is of great importance in planar systems. To differentiate observing similar cases, some researchers introduce boundary information for plane features [16], [17], [18]. However, this suffers from the partial observability problem that planes in the real world may be obscured by cluttered objects or out of the current camera view. The plane shape, area, and center change incrementally, which brings challenges for data association. Others attempt to jointly associate a group of planes by considering their relations or combine objects to increase the distinguishability of scenes [19], [40], [41].

In order to gain a deep understanding of surroundings and estimate the robot's state intelligently, some methods aim to represent the environments using high-level geometric elements that integrate structural, semantic as well as topological constraints of planes. Bavle et al. [20] proposed a three-layered situational graph that uses corridors and rooms to constrain planar walls and robot poses in structured indoor environments. Shaheer et al. [42] extended it to an architectural graph for building information modeling by introducing high-level walls, doorways, and floors to constrain wall surfaces and rooms, and applied a graph-to-graph matching method to achieve global robot localization. In [15], right corner features based on orthogonal planes first came into the view. Similar to the MW assumption, the orientation of the right corner is aligned with a global coordinate system called the building coordinate system, and is not applied in the matching process, which is not suitable for typical real-world scenarios. In addition, Sommer et al. [43] proposed a joint corner and plane extraction method that demonstrates the ability to use corners for scan alignment and robot positioning.

Inspired by the above research, in this work, we model the plane boundaries using convex hulls and propose box corners within a VINS, to help with robust data association as well as accurate localization and mapping. Different from those prior assumptions, box corners are more general and can be utilized to

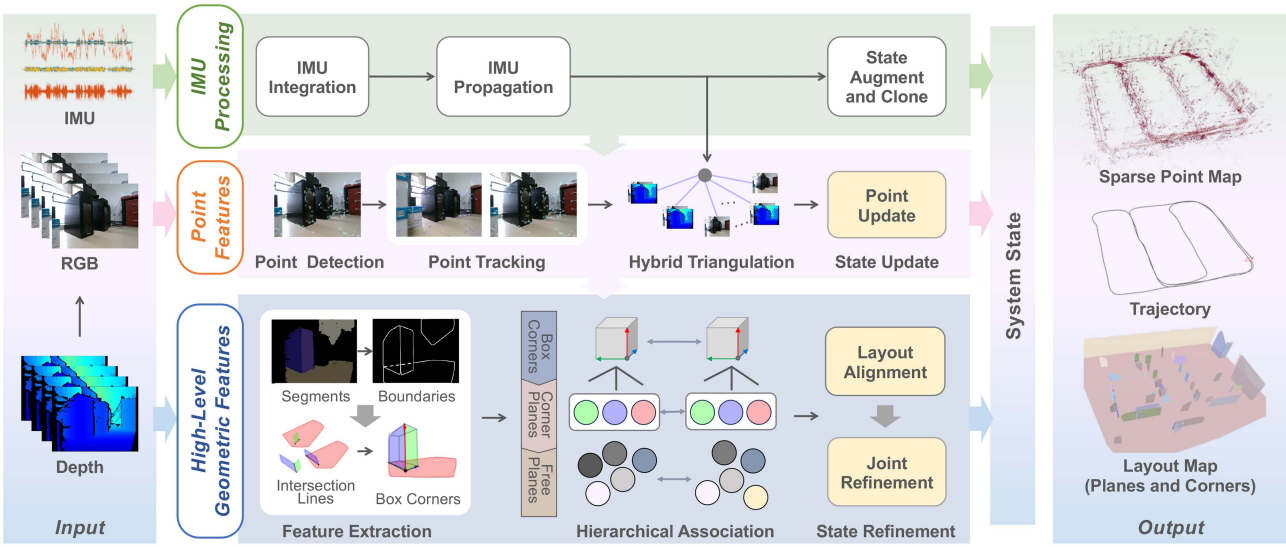

Fig. 2. Overview of the proposed system, where depth information is utilized to enhance the point-based system, upon which high-level plane and corner features further refine the system state.

compute the drift-free rotation and translation simultaneously. More notably, they integrate planar structural regularities and can be efficiently extracted and associated, enabling the understanding of an indoor environment at a high level.

III. System Overview

CornerVINS takes RGB-D images and IMU data as input and adopts a lightweight EKF framework that utilizes hierarchical geometric features including points, planes, and box corners. The system contains three key components: 1) IMU processing; 2) state update with depth-enhanced point features; and 3) state refinement with high-level geometric features, as shown in Fig. 2. IMU processing is responsible for the system initialization and EKF state propagation between adjacent camera frames. Subsequently, point features are employed in the EKF state update after triangulation to provide the initial camera pose. Afterward, the system state is further refined using high-level plane and corner features, which jointly consist of a layout map.

A. System State

The state vector of our system is defined as

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^\mathsf{T} & \mathbf{x}_{\text{calib}}^\mathsf{T} & \mathbf{x}_C^\mathsf{T} & \mathbf{x}_P^\mathsf{T} & \mathbf{x}_\Pi^\mathsf{T} & \mathbf{x}_B^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{1}$$

$\mathbf{x}_{I_k}$ is the IMU state at time step k, which can be represented as

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{\mathbf{q}}^\mathsf{T} & {}^{G}\mathbf{p}_{I_k}^\mathsf{T} & {}^{G}\mathbf{v}_{I_k}^\mathsf{T} & {}^{I_k}\mathbf{b}_g^\mathsf{T} & {}^{I_k}\mathbf{b}_a^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{2}$$

where ${}^{I_k}_{G}\bar{\mathbf{q}}$ is the unit quaternion representing the rotation from the world frame $\{G\}$ to the IMU frame $\{I_k\}$, ${}^{G}\mathbf{p}_{I_k}$ and ${}^{G}\mathbf{v}_{I_k}$ are the position and the velocity of the IMU with respect to $\{G\}$, ${}^{I_k}\mathbf{b}_g$ and ${}^{I_k}\mathbf{b}_a$ are biases of the gyroscope and the accelerometer, respectively, $\mathbf{x}_{\text{calib}}$ is calibration parameters, including camera-IMU rigid transformation $\{{}^{C}_{I}\bar{\mathbf{q}},\ {}^{C}\mathbf{p}_I\}$, time offset ${}^{C}_{I}t$, and camera intrinsic parameters $\boldsymbol{\lambda}_C$, namely,

$$\mathbf{x}_{\text{calib}} = \begin{bmatrix} {}^{C}_{I}\bar{\mathbf{q}}^\mathsf{T} & {}^{C}\mathbf{p}_I^\mathsf{T} & {}^{C}_{I}t & \boldsymbol{\lambda}_C^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{3}$$

$\mathbf{x}_C$ indicates historical IMU pose clones in the sliding window, with the form of

$$\mathbf{x}_C = \begin{bmatrix} {}^{I_{k-c}}_{G}\bar{\mathbf{q}}^\mathsf{T} & {}^{G}\mathbf{p}_{I_{k-c}}^\mathsf{T} & \dots & {}^{I_k}_{G}\bar{\mathbf{q}}^\mathsf{T} & {}^{G}\mathbf{p}_{I_k}^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{4}$$

where $[{}^{I_k}_{G}\bar{\mathbf{q}}\ {}^{G}\mathbf{p}_{I_k}]$ is the IMU pose clone at time step k, and c is the size of the window.

Following the filter structure in [3], some features are also involved in the state vector for further refinement. $\mathbf{x}_P$, $\mathbf{x}_\Pi$, and $\mathbf{x}_B$ are sets of h point, n plane, and m box corner features in the world coordinate system, respectively. They are expressed as

$$\mathbf{x}_P = \begin{bmatrix} {}^{G}\mathbf{f}_1^\mathsf{T} & {}^{G}\mathbf{f}_2^\mathsf{T} & \dots {}^{G}\mathbf{f}_h^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{5}$$

$$\mathbf{x}_\Pi = \begin{bmatrix} {}^{G}\boldsymbol{\Pi}_1^\mathsf{T} & {}^{G}\boldsymbol{\Pi}_2^\mathsf{T} & \dots {}^{G}\boldsymbol{\Pi}_n^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{6}$$

and

$$\mathbf{x}_B = \begin{bmatrix} {}^{G}\mathbf{B}_1^\mathsf{T} & {}^{G}\mathbf{B}_2^\mathsf{T} & \dots {}^{G}\mathbf{B}_m^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{7}$$

The representation of each feature will be introduced in the following sections.

B. Notation

Throughout this article, ${}^{B}_{A}\mathbf{R}$ is the matrix form of the unit quaternion ${}^{B}_{A}\bar{\mathbf{q}}$, representing the rotation from frame $\{A\}$ to frame $\{B\}$, and ${}^{B}\mathbf{p}_A$ is a vector representing the translation from frame $\{A\}$ to frame $\{B\}$. The camera pose relative to $\{G\}$ is computed by

$$\begin{cases} {}^{C}_{G}\mathbf{R} &= {}^{C}_{I}\mathbf{R}\,{}^{I}_{G}\mathbf{R} \\ {}^{G}\mathbf{p}_C &= {}^{G}\mathbf{p}_I - {}^{I}_{G}\mathbf{R}^\mathsf{T}\,{}^{C}_{I}\mathbf{R}^\mathsf{T}\,{}^{C}\mathbf{p}_I. \end{cases} \tag{8}$$

For simplicity, we assume that the IMU frame and the RGB-D camera frame coincide as frame $\{C\}$ in subsequent expressions.

IV. Depth Enhanced Point Features

We additionally incorporate depth information into the point feature triangulation and state update process based on [3].

A. IMU Processing

1) System Initialization: The system state in (2) is initialized using IMU measurements collected during a stationary period before motion [3]. In the stationary window, the accelerometer readings reflect the opposite direction of gravity. Hence, the initial IMU pose ${}^{I_0}_{G}\bar{\mathbf{q}}$ and ${}^{G}\mathbf{p}_{I_0}$ are computed by aligning the z-axis of the world coordinate system with gravity, while keeping the origin coincident with the IMU coordinate system. The initial velocity ${}^{G}\mathbf{v}_{I_0}$ is set to zero. Furthermore, the accelerometer and gyroscope biases ${}^{I_0}\mathbf{b}_g$ and ${}^{I_0}\mathbf{b}_a$ are initialized using their average readings during the window, after eliminating the interference of gravity.

2) Integration and Propagation: Before the feature update, the system state evolves from time t_k to t_{k+1} through IMU integration

$$\mathbf{x}_{I_{k+1}} = \mathcal{I}(\mathbf{x}_{I_k}, \mathbf{a}_{k+1|k}, \mathbf{g}_{k+1|k}) \tag{9}$$

where $\mathcal{I}(\cdot)$ is the integration function. $\mathbf{a}_{k+1|k}$ and $\mathbf{g}_{k+1|k}$ are linear acceleration and angular velocity measurements between time step k and $k+1$, which are contaminated by zero-mean white Gaussian noises and random walk biases. For more details about the generic nonlinear IMU kinematics and the state evolution process see [1] and [3].

B. Point Detection and Tracking

We extract hybrid FAST [44] keypoints with and without depth from input RGB-D images uniformly by dividing the images into several separate grids. A point measurement is represented as $\mathbf{f}_{\text{obs}} = (u, v, z)$, where (u, v) is the pixel coordinates in the color image C_k and z is the depth value at (u, v) that can be obtained from the aligned depth image D_k or is set to -1 when it is not available. The corresponding 3-D point is represented as $\mathbf{f} = (X, Y, Z)$.

The sparse kanade-lucas-tomasi (KLT) optical flow algorithm [45] is employed to track 2-D keypoints between successive frames. When tracking is lost, the point is considered to have disappeared from the field of view, and the observation is no longer generated. Keypoints that are not successfully tracked from the previous frame are treated as new features.

C. Hybrid Point Triangulation

To preliminarily estimate the 3-D positions of point features, we take historical camera poses in the sliding window and the current camera pose predicted by the IMU propagation to be of known quantities and fuse depth information into the original 3-D Cartesian triangulation. In particular, when a point ${}^{G}\mathbf{f}$ is observed by a camera C_m, we have an observation

$${}^{C_m}\mathbf{f} = {}^{C_m}z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = {}^{C_m}z\,{}^{C_m}\mathbf{b}_f \tag{10}$$

where ${}^{C_m}z$ represents the depth of this point from the image plane and ${}^{C_m}\mathbf{b}_f$ is the bearing vector. With the knowledge of ${}^{C_m}z$, we can directly transform the 3-D observation ${}^{C_m}\mathbf{f}$ to the world frame $\{G\}$ via

$${}^{G}\mathbf{f} = {}^{C_m}_{G}\mathbf{R}^{\top}{}^{C_m}\mathbf{f} + {}^{G}\mathbf{p}_{C_m}. \tag{11}$$

Otherwise, if ${}^{C_m}z$ is not available due to noise or exceeding the range, ${}^{G}\mathbf{f}$ can be written as

$$\begin{aligned} {}^{G}\mathbf{f} &= {}^{C_m}_{G}\mathbf{R}^{\top}{}^{C_m}z\,{}^{C_m}\mathbf{b}_f + {}^{G}\mathbf{p}_{C_m} \\ &= {}^{C_m}z\,{}^{G}\mathbf{b}_f + {}^{G}\mathbf{p}_{C_m}. \end{aligned} \tag{12}$$

By defining vectors orthogonal to ${}^{G}\mathbf{b}_f$ in $\mathbf{N}_m$ (${\mathbf{N}_m}^{G}\mathbf{b}_f = \mathbf{0}_{3\times 3}$) [3]

$$\begin{aligned} \mathbf{N}_m &= \lfloor {}^{G}\mathbf{b}_f \times \rfloor \\ &= \begin{bmatrix} 0 & {}^{G}\mathbf{b}_f(2) & {}^{G}\mathbf{b}_f(1) \\ {}^{G}\mathbf{b}_f(2) & 0 & {}^{G}\mathbf{b}_f(0) \\ {}^{G}\mathbf{b}_f(1) & {}^{G}\mathbf{b}_f(0) & 0 \end{bmatrix} \end{aligned} \tag{13}$$

and substituting it to (12), we can obtain

$$\begin{aligned} \mathbf{N}_m{}^{G}\mathbf{f} &= \mathbf{N}_m{}^{C_m}z\,{}^{G}\mathbf{b}_f + \mathbf{N}_m{}^{G}\mathbf{p}_{C_m} \\ &= \mathbf{N}_m{}^{G}\mathbf{p}_{C_m}. \end{aligned} \tag{14}$$

After stacking all the hybrid observations (2-D and 3-D)

$$\underbrace{\begin{bmatrix} \vdots \\ \mathbf{N}_{m_1} \\ \vdots \\ \mathbf{I}_{3\times 3} \\ \vdots \end{bmatrix}}_{\mathbf{A}} {}^{G}\mathbf{f} = \underbrace{\begin{bmatrix} \vdots \\ \mathbf{N}_{m_1}{}^{G}\mathbf{p}_{C_{m_1}} \\ \vdots \\ {}^{C_{m_2}}_{G}\mathbf{R}^{\top}{}^{C_{m_2}}\mathbf{f} + {}^{G}\mathbf{p}_{C_{m_2}} \\ \vdots \end{bmatrix}}_{\mathbf{b}} \tag{15}$$

the 3-D point feature ${}^{G}\mathbf{f}$ is calculated by solving the linear system

$$\mathbf{A}^{\top}\mathbf{A}\,{}^{G}\mathbf{f} = \mathbf{A}^{\top}\mathbf{b}. \tag{16}$$

D. Point Measurement Model

In addition, we extend the standard point measurement model to include another 1-D depth measurement. Typically, a point feature ${}^{G}\mathbf{f}$ is updated using the following measurement function [1], [3]:

$$\mathbf{z}_b = h(\mathbf{x}_k) + \mathbf{n}_b \tag{17}$$

where $h(\cdot)$ projects ${}^{G}\mathbf{f}$ onto an observed image C_m with state $\mathbf{x}_k$, and $\mathbf{n}_b \sim \mathcal{N}(\mathbf{0}_{2\times 2}, \mathbf{I}_{2\times 2})$ denotes the measurement noise.

For points with valid depths, the 1-D depth measurement is introduced by

$$z_d = \mathbf{\Lambda}_G^{C_m}\mathbf{R}({}^G\mathbf{f} - {}^G\mathbf{p}_{C_m}) + n_d,\ \mathbf{\Lambda} = \begin{bmatrix}0\ 0\ 1\end{bmatrix} \tag{18}$$

where $n_d \sim \mathcal{N}(0, {\sigma_d}^2)$ represents the depth measurement noise with $\sigma_d = \alpha z_d$ ($\alpha = 4\%$ in our experiments).

Linearizing the equations yields the following system [1], [3]:

$$\begin{bmatrix}\tilde{\mathbf{z}}_b \\ \tilde{z}_d\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{T_b} \\ \mathbf{H}_{T_d}\end{bmatrix}\tilde{\mathbf{x}}_{T_m} + \begin{bmatrix}\mathbf{H}_{f_b} \\ \mathbf{H}_{f_d}\end{bmatrix}{}^G\tilde{\mathbf{f}} + \begin{bmatrix}\mathbf{n}_b \\ n_d\end{bmatrix}$$

$$\Rightarrow \tilde{\mathbf{z}}_m = \mathbf{H}_{T_m}\tilde{\mathbf{x}}_{T_m} + \mathbf{H}_{f_m}{}^G\tilde{\mathbf{f}} + \mathbf{n}_m \tag{19}$$

where $\tilde{\mathbf{z}}_b$ is the projected 2-D residual, $\tilde{z}_d$ is the 1-D depth residual, and $\mathbf{n}_b$ and n_d are white Gaussian noises. $\mathbf{H}_{T_b}$ and $\mathbf{H}_{T_d}$ are the measurement Jacobians in respect to the current state $\mathbf{x}_{T_m}$. Similarly, $\mathbf{H}_{f_b}$ and $\mathbf{H}_{f_d}$ are the measurement Jacobians in respect to the 3-D point feature ${}^G\mathbf{f}$.

Stacking all the hybrid measurements from different timesteps leads to the following measurement function for a point ${}^G\mathbf{f}$:

$$\tilde{\mathbf{z}} = \mathbf{H}_T\tilde{\mathbf{x}}_T + \mathbf{H}_f{}^G\tilde{\mathbf{f}} + \mathbf{n} \tag{20}$$

where $\tilde{\mathbf{z}}$ is the stacked residual including projected 2-D residuals and 1-D depth residuals if valid, $\mathbf{H}_T$ is the stacked Jacobian for camera poses $\mathbf{x}_T$, $\mathbf{H}_f$ is the stacked Jacobian for point feature ${}^G\mathbf{f}$, and $\mathbf{n}$ denotes the stacked measurement noise.

E. State Update and Sparse Mapping

The point-based state update process provides not only initial camera pose estimations of input images but also sparse 3-D points. The problem is formulated as an EKF update using point measurement model (20).

In our system, points are divided into multi-state constraint kalman filter (MSCKF) features and SLAM features based on their track lengths [3]. We record the number of frames where each point has been continuously observed. If the number is larger than the window size, the point is classified as a stable SLAM feature and added to the state vector for refinement. Otherwise, it is considered as a short-lived MSCKF feature. When a new frame comes, all measurements of new MSCKF features, i.e., points lost on this frame, and all new measurements of SLAM features participate in the update. Since only SLAM features are in the state vector, SLAM points are updated using standard EKF, while feature dependency will be removed from (20) through nullspace projection for MSCKF points [1], [3]. After the update, the 3-D coordinates of the MSCKF points no longer change because no observations will be generated, and the SLAM points will be continuously updated until they are marginalized from the state vector when tracking is lost. Together, they incrementally form a sparse 3-D point map.

V. High-Level Geometric Features

To fully explore structural information in indoor environments, we propose a distinctive feature, namely, box corner, and fuse it with the plane feature, providing high-level constraints to enhance the performance of the above point-based system. The planes and box corners are extracted and associated effectively using a hierarchical mechanism. They provide a compact representation of the indoor scene and jointly consist of a layout map.

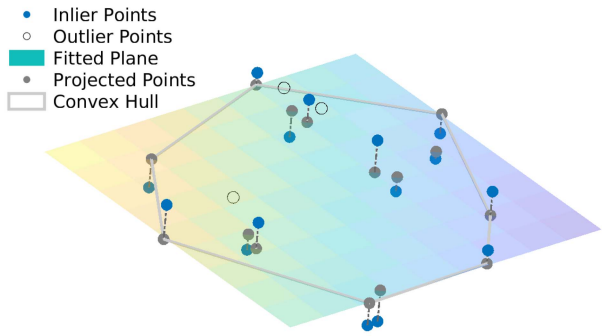

Fig. 3. Illustration of computing the convex hull for a plane.

A. Plane Extraction

In our case, a plane is modeled as a bounded patch using convex hulls. We detect planes from input depth images in four steps: 1) planar region segmentation; 2) parameter fitting; 3) plane merging; and 4) parameter refitting. First, the agglomerative hierarchical clustering (AHC) method [46] is chosen to segment planar regions from structured point clouds, which can effectively extract planes with irregular shapes and in noisy environments. To ensure the efficiency and accuracy, planar segmentation regions with fewer than 20 000 points are discarded. After downsampling the point clouds of the remaining planes, we fit the initial parameters by minimizing the point-to-plane distances with RANSAC, and retain planes with at least 90% inliers. Moreover, similar planes with small angles ($< 5^\circ$) and distances (< 0.02 m) will be iteratively merged into a large one. Finally, we refit new parameters for the merged planes and compute their 3-D convex hulls after projecting all the inlier points onto planes, as illustrated in Fig. 3. This boundary modeling method increases the distinction between planes using only a few points and lays the foundation for the subsequent extraction of actual intersections.

B. Box Corner Extraction

In indoor environments, three mutually orthogonal planes form a stable physical structure, which we regard as a box corner. It encodes rich geometric and structural information to understand the scene at a high level, exhibiting strong robustness and distinguishability for robot localization and mapping. A box corner can be represented by a local coordinate system associated with a box region. We find this particular area by checking the recognized planes in current frame according to the process outlined in Algorithm 1.

Considering three approximately orthogonal planes i, j, and k (lines 2–4), their intersection ${}^C\mathbf{p}_B$ can be computed by solving (line 5)

$$\mathbf{N}{}^C\mathbf{p}_B = \mathbf{D} \tag{21}$$

Algorithm 1: Box Corner Extraction.

Input: detected planes $\{\mathbf{\Pi}\}$
Output: detected box corners $\{\mathbf{B}\}$

1: **for** plane $\mathbf{\Pi}_i, \mathbf{\Pi}_j, \mathbf{\Pi}_k \in \{\mathbf{\Pi}\}$ **do**
2: **if** $\theta_{ij} < \theta_\delta$ or $\theta_{ik} < \theta_\delta$ or $\theta_{jk} < \theta_\delta$ **then**
3: continue; /* θ_δ is the angle threshold that determines whether two planes are orthogonal. */
4: **end if**
5: $\mathbf{p}_B = IntersectionOfThreePlanes(\mathbf{\Pi}_i, \mathbf{\Pi}_j, \mathbf{\Pi}_k)$;
6: $[\mathbf{g}_{ij}, \mathbf{p}_{ij}] = EdgeOfTwoPlanes(\mathbf{\Pi}_i, \mathbf{\Pi}_j, \mathbf{p}_B)$;
7: $[\mathbf{g}_{ik}, \mathbf{p}_{ik}] = EdgeOfTwoPlanes(\mathbf{\Pi}_i, \mathbf{\Pi}_k, \mathbf{p}_B)$;
8: $[\mathbf{g}_{jk}, \mathbf{p}_{jk}] = EdgeOfTwoPlanes(\mathbf{\Pi}_j, \mathbf{\Pi}_k, \mathbf{p}_B)$;
9: **if** $Distance(\mathbf{p}_B, \mathbf{g}_{ij}) > d_\delta$ or $Distance(\mathbf{p}_B, \mathbf{g}_{ik}) > d_\delta$ or $Distance(\mathbf{p}_B, \mathbf{g}_{jk}) > d_\delta$ **then**
10: continue; /* d_δ is the maximum acceptable distance between the corner point and the closer endpoint of the intersection line segment. */
11: **end if**
12: **if** $Distance(\mathbf{p}_B, \mathbf{p}_{ij}) < l_\delta$ or $Distance(\mathbf{p}_B, \mathbf{p}_{ik}) < l_\delta$ or $Distance(\mathbf{p}_B, \mathbf{p}_{jk}) < l_\delta$ **then**
13: continue; /* l_δ is the minimum acceptable length of the box edge. */
14: **end if**
15: $\mathbf{v}_k = (\mathbf{p}_{ij} - \mathbf{g}_{ij})/||\mathbf{p}_{ij} - \mathbf{g}_{ij}||$;
16: $\mathbf{v}_j = (\mathbf{p}_{ik} - \mathbf{g}_{ik})/||\mathbf{p}_{ik} - \mathbf{g}_{ik}||$;
17: $\mathbf{v}_i = (\mathbf{p}_{jk} - \mathbf{g}_{jk})/||\mathbf{p}_{jk} - \mathbf{g}_{jk}||$;
18: $\mathbf{x} = \mathbf{v}_k \times \mathbf{v}_j$;
19: **if** $\mathbf{x}^\mathsf{T}\mathbf{v}_i < 0$ **then**
20: $[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \mathrm{SVD}([\mathbf{v}_i\ \mathbf{v}_k\ \mathbf{v}_j])$;
21: **else**
22: $[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \mathrm{SVD}([\mathbf{v}_i\ \mathbf{v}_j\ \mathbf{v}_k])$;
23: **end if**
24: ${}^C\mathbf{p}_B = \mathbf{p}_B$, ${}^C_B\mathbf{R} = \mathbf{U}\mathbf{V}^\mathsf{T}$
25: **end for**
26: **return** $\{\mathbf{B} = [{}^C_B\mathbf{R}\ {}^C\mathbf{p}_B]\}$

where

$$\mathbf{N} = \begin{bmatrix} \mathbf{n}_i^\mathsf{T} \\ \mathbf{n}_j^\mathsf{T} \\ \mathbf{n}_k^\mathsf{T} \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} d_i \\ d_j \\ d_k \end{bmatrix}, \quad {}^C\mathbf{p}_B = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{22}$$

To ensure that ${}^C\mathbf{p}_B$ is a nonvirtual intersection in the scene, we compute the intersection line segments between each pair of plane convex hulls (lines 6–8) and ignore those corners that are formed by virtual intersection lines or too far from the closer endpoints ($\mathbf{g}_{ij}$, $\mathbf{g}_{ik}$ and $\mathbf{g}_{jk}$) (lines 9–11). In addition, corners supported by short intersection line segments are discarded because they are unstable (lines 12–14). This process is depicted in Fig. 4. More specifically, after calculating the three infinite intersection lines of the plane triplet, we identify the valid segments of the intersection line and the convex hull for each plane, in their respective 2-D spaces. A geometric tolerance range is set to mitigate the influence of noise. The intersection points of the plane convex hull and this range, along with all vertices in it, determine the valid length of the intersection line [see Fig. 4(a)].

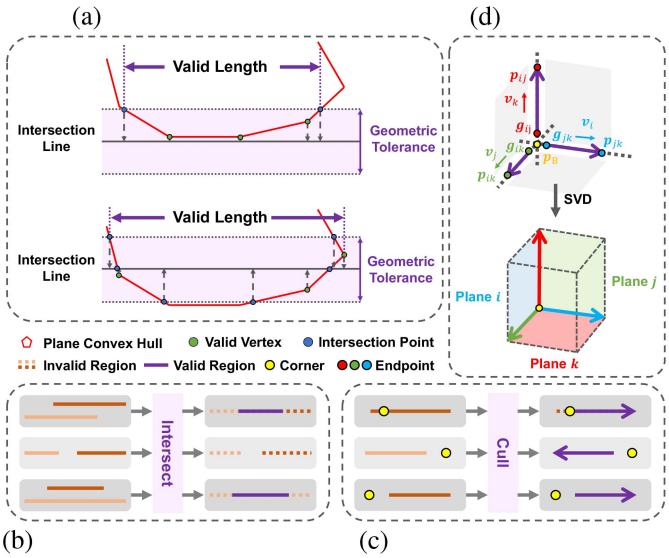

Fig. 4. Diagram of calculating the box corner. (a) Two cases of the plane convex hull and the intersection line, both taking the union of the points to determine the valid length. If no valid vertex or intersection point is found, the line is considered a virtual intersection that does not satisfy the conditions for forming a reliable corner. (b) Common valid area of two intersection line segments. After calculating the respective intersection line segments in (a), the two segments of each pair of planes are merged into one. (c) Final valid region with direction (purple arrow), after culling out the outside part based on (b). (d) Endpoints and direction vectors of the extracted corner, forming a box region.

The final edge of the two plane patches is established by the intersection set of their line segments, culling out the portions outside the corner box [see Fig. 4(b) and (c)].

Then, the direction vectors of these intersection lines ($\mathbf{v}_k$, $\mathbf{v}_j$ and $\mathbf{v}_i$) are computed, pointing from the closer endpoints ($\mathbf{g}_{ij}$, $\mathbf{g}_{ik}$, and $\mathbf{g}_{jk}$) to the farther ($\mathbf{p}_{ij}$, $\mathbf{p}_{ik}$, and $\mathbf{p}_{jk}$) (lines 15–17). Due to the presence of noise, they are not completely perpendicular, so we project them onto the $\mathrm{SO}(3)$ manifold to satisfy the orthogonality constraint using singular value decomposition (SVD) (line 24). The refined directions are expressed as a rotation matrix ${}^C_B\mathbf{R}$, where each column corresponds to the x, y, and z-axis of the corner coordinate system $\{B\}$ with respect to the camera coordinate system $\{C\}$, and obeys the right hand rule (lines 18–24).

Therefore, the box corner coordinate system is established with the intersection point ${}^C\mathbf{p}_B$ as the origin and the columns of the rotation ${}^C_B\mathbf{R}$ as the three axes. Moreover, the intersection line segments of the plane patches are used to construct a box region, which indicates the range where the corner is observed. The box provides an adaptive distance threshold for comparing corner positions, which is less sensitive to accumulated errors in the initial pose and will not be used in optimization.

Specially, we classify two types of box corners as in Fig. 5, which can be distinguished by comparing the orientation to the camera

$$\mathbf{n}_B = {}^C\mathbf{p}_B + \mathbf{v}_i + \mathbf{v}_j + \mathbf{v}_k. \tag{23}$$

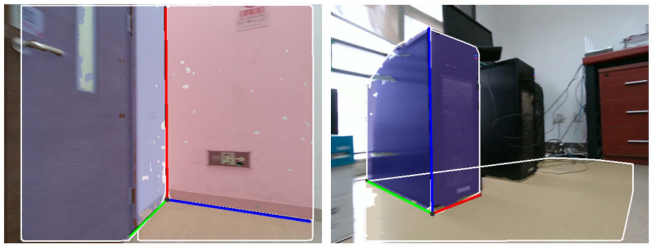

Fig. 5. Two kinds of box corners. (a) Corner observed from the inside of the box (IBC). (b) Corner observed from the outside of the box (OBC). The red, green, and blue lines are the x, y, and z-axis of the box corner, respectively, and the lengths of the three lines represent the edges of the box.

If $\mathbf{n}_B \cdot \mathbf{e}_3 < 0$, the corner is observed from the inside of the box and will be marked as an IBC. Otherwise, it is observed from the outside of the box and will be marked as an OBC.

C. Hierarchical Data Association

Typically, detected planes are transformed to the global coordinate system using initial poses and then associated with historical planes based on the difference in their normal vectors and distances. However, the method becomes unstable when the poses drift due to error accumulation, especially in planar ambiguous regions. The box corner that combines three planes while seamlessly integrating structural constraints is more informative and distinguishable than a single plane. We propose a hierarchical data association strategy for corners and planes to increase efficiency and robustness.

1) Box Corner Association: Using initial pose ${}^C_G\mathbf{R}$ and ${}^G\mathbf{p}_C$ obtained from the point feature update, we transform the position ${}^C\mathbf{p}_B$ of the detected box corner $\mathbf{B}$ to the world coordinate system $\{G\}$ via

$$ {}^G\mathbf{p}_B = {}^C_G\mathbf{R}^{\mathsf{T}} {}^C\mathbf{p}_B + {}^G\mathbf{p}_C. \tag{24}$$

Since the x, y, and z-axis of the corner coordinate system are randomly established each time on the basis of satisfying the right hand rule, a rotation needs to be determined to correspond the three axes accordingly when associated with corner landmarks. Let $\{B\}$ be the detected box corner frame, $\{B'\}$ be the aligned box corner frame, then the rotation ${}^B_{B'}\mathbf{R}$ belongs to one of the following three matrices:

$$ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \text{and} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{25}$$

Therefore, the corner orientation in $\{G\}$ is represented as

$$ {}^G_{B'}\mathbf{R} = {}^C_G\mathbf{R}^{\mathsf{T}} {}^C_B\mathbf{R} {}^B_{B'}\mathbf{R}. \tag{26}$$

Then, two corners ${}^C\mathbf{B} = [{}^C_B\mathbf{R}\ {}^C\mathbf{p}_B]$ and ${}^G\hat{\mathbf{B}} = [{}^G_B\hat{\mathbf{R}}\ {}^G\hat{\mathbf{p}}_B]$ are matched if their positions in $\{G\}$ are close and their orientation error ϵ is within the threshold ($< 15^\circ$), which is defined as

$$ \epsilon = \arccos\left(\frac{\mathrm{tr}({}^G_{B'}\mathbf{R}^{\mathsf{T}} {}^G_B\hat{\mathbf{R}}) - 1}{2}\right). \tag{27}$$

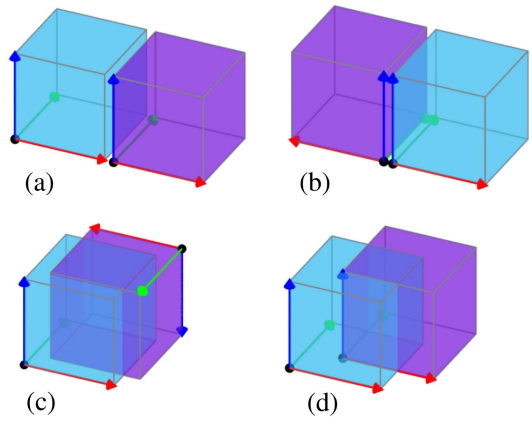

Fig. 6. Four spatial configurations of two box corners. (a) Similar in orientation but distinct in position. (b) Similar in position but distinct in orientation. (c) Distinct in both position and orientation. (d) Similar in both position and orientation. Only in the last case, the two corners will be associated.

Accordingly, the edges of the box are expanded. The way to determine whether two corners are close in position is to compare the spatial relationship of their boxes. The robust corner association is facilitated by the fact that the boxes of two corners in the map should not overlap too much to block the robot's view. Otherwise, they should be regarded as the same physical entity. Experimentally, box corners are spatially dispersed, with superior uniqueness and distinguishability by comparing the above mentioned properties. They are either in different directions or at far distances due to the limitations of the camera field of view, depth perception range, and the robot's own volume. Fig. 6 describes four spatial configurations of two box corner pairs. The box area provides an adaptive way to associate corners. Note that different kinds of box corners (IBC and OBC) will not be associated together.

Once a box corner is associated successfully, the three planes related to it (corner planes) are automatically assigned to mapped planes. It allows for very robust and fast data association, resulting in more accurate 3-D maps.

2) Plane Association: We use the following four criteria for the association of the rest planes (free planes): a) difference between plane normals; b) distance; c) projection overlapping; and d) separation between plane patches. The angle of two plane normals is computed by

$$ \theta = \arccos\left(\mathbf{n}_i, \mathbf{n}_j\right), \ \left(0 \le \theta \le \frac{\pi}{2}\right). \tag{28}$$

As infinite planes are not that much distinguishable, boundary information is considered in the association to increase the uniqueness of planes. In particular, the distance l between two plane patches (i, j) is defined as

$$ l = \max\{l_{ij}, l_{ji}\} \tag{29}$$

$$ l_{ij} = \begin{cases} 0, & \text{if } \Delta \le 0 \\ \min\limits_{\mathbf{p}_{ik} \in \mathcal{H}_i} |l(\mathbf{p}_{ik}, \mathbf{\Pi}_j)|, & \text{otherwise} \end{cases} \tag{30}$$

$$ \Delta = \max_{\mathbf{p}_{ik} \in \mathcal{H}_i} l(\mathbf{p}_{ik}, \mathbf{\Pi}_j) \times \min_{\mathbf{p}_{ik} \in \mathcal{H}_i} l(\mathbf{p}_{ik}, \mathbf{\Pi}_j) \tag{31}$$

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

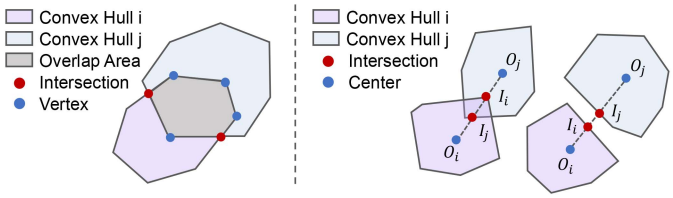

Fig. 7. Illustration of the overlap (left) and the separation (right) of two convex hulls on the projected plane (reference plane).

where $\mathbf{p}_{ik}$ is a convex hull point of plane i, $l(\cdot,\cdot)$ is the directed point-to-plane distance, l_{ij} is the minimum distance from all convex points on plane i to plane j or zero (when the convex hull points are on both sides of the plane), and the opposite for l_{ji}.

In addition, if their normals are close, we project the two convex hulls on a reference plane, i.e., the average plane of the two planes, and compare their overlapping and separation. As illustrated in Fig. 7, the overlap area of two planes is supported by the intersections and vertices of the convex hulls and the separation s is computed by

$$s = \max\{0, s(O_i, O_j) - s(O_i, I_i) - s(O_j, I_j)\} \tag{32}$$

where O_i and O_j are the center points of the two projected convex hulls, I_i and I_j are the intersections of their connection line and the two convex hulls, and $s(\cdot,\cdot)$ is the Euclidean distance of two points.

Only if all four conditions are within the preset thresholds, the two planes are considered to belong to the same one. Furthermore, to improve the accuracy and efficiency of the plane association, we maintain a local map containing planes that disappear for less than ten frames, for priority candidate search.

D. Plane Measurement Model

In the state vector, a plane is represented in the minimal parameterization using the closest point (CP) form [47]

$$\mathbf{\Pi} = \mathbf{n}d, \quad \begin{bmatrix}\mathbf{n}\\ d\end{bmatrix} = \begin{bmatrix}\mathbf{\Pi}/||\mathbf{\Pi}||\\ ||\mathbf{\Pi}||\end{bmatrix} \tag{33}$$

where $\mathbf{n}$ and d are the unit normal vector and the distance scalar of the plane, respectively. This representation treats the plane as the CP to the reference coordinate system, allowing a simple additive error model and free of singularities in our system.

As environmental planes can be directly extracted from RGB-D cameras, we establish the measurement model for plane feature ${}^{G}\mathbf{\Pi}$ based on the similarity transformation

$$\begin{aligned}{}^{C_m}\mathbf{\Pi} &= {}^{C_m}\mathbf{n}\,{}^{C_m}d \\ &= ({}^{C_m}_{G}\mathbf{R}\,{}^{G}\mathbf{n})({}^{G}d - {}^{G}\mathbf{p}_{C_m}^{\mathsf{T}}\,{}^{G}\mathbf{n}) + \mathbf{n}_m^{\Pi}\end{aligned} \tag{34}$$

where ${}^{C_m}_{G}\mathbf{R}$ and ${}^{G}\mathbf{p}_{C_m}$ are the pose of camera C_m, $\mathbf{n}_m^{\Pi}$ is zero-mean Gaussian noise, whose covariance is computed based on its inlier points [13]. We linearize this equation and obtain the following residual and Jacobians [47]:

$$\tilde{\mathbf{\Pi}}_m = \mathbf{H}_{T_m}\tilde{\mathbf{x}}_{T_m} + \mathbf{H}_{\mathbf{\Pi}_m}\,{}^{G}\tilde{\mathbf{\Pi}} + \mathbf{n}_m^{\Pi} \tag{35}$$

in which $\tilde{\mathbf{\Pi}}_m$ is the 3-D plane measurement residual, $\mathbf{H}_{T_m}$ and $\mathbf{H}_{\mathbf{\Pi}_m}$ are the plane measurement Jacobians with respect to the camera pose $\mathbf{x}_{T_m}$ and the plane feature ${}^{G}\mathbf{\Pi}$, respectively. After stacking all the new measurements, we obtain the residual model for plane ${}^{G}\mathbf{\Pi}$

$$\tilde{\mathbf{\Pi}} = \mathbf{H}_T\tilde{\mathbf{x}}_T + \mathbf{H}_{\mathbf{\Pi}}\,{}^{G}\tilde{\mathbf{\Pi}} + \mathbf{n}^{\Pi} \tag{36}$$

where $\tilde{\mathbf{\Pi}}$ is the stacked plane residual, $\mathbf{H}_T$ and $\mathbf{H}_{\mathbf{\Pi}}$ are stacked Jacobians, and $\mathbf{n}^{\Pi}$ is the stacked measurement noise.

E. Corner Measurement Model

A box corner found on the intersection of three orthogonal planes has 6 DoF, including a 3-DoF position and a 3-DoF orientation. It can be observed by the following transformation:

$$\begin{cases}{}^{C_m}_{B}\mathbf{R} = {}^{C_m}_{G}\mathbf{R}\,{}^{G}_{B'}\mathbf{R}\,{}^{B}_{B'}\mathbf{R}^{\mathsf{T}} \\ {}^{C_m}\mathbf{p}_B = {}^{C_m}_{G}\mathbf{R}({}^{G}\mathbf{p}_B - {}^{G}\mathbf{p}_{C_m}).\end{cases} \tag{37}$$

${}^{C_m}_{G}\mathbf{R}$ and ${}^{G}\mathbf{p}_{C_m}$ are the camera pose. ${}^{G}\mathbf{p}_B$ and ${}^{C_m}\mathbf{p}_B$ are corner positions in the world and the camera coordinate system, respectively. ${}^{G}_{B}\mathbf{R}$ and ${}^{C_m}_{B}\mathbf{R}$ are the rotation matrix from the corner coordinate system to the world and the camera coordinate system, respectively, and ${}^{B}_{B'}\mathbf{R}$ is obtained during data association, as described in Section V-C. In particular, we use the unit quaternion to minimally parameterize the rotation of a box corner in the system state vector. The standard additive error and the left quaternion multiplicative error are employed to define the error state for the position and the rotation part of the box corner, respectively. Based on the small angle assumption, the quaternion error $\delta\bar{\mathbf{q}}$ is approximated as

$$\delta\bar{\mathbf{q}} = \begin{bmatrix}\hat{\mathbf{k}}\sin(\frac{1}{2}\tilde{\theta})\\ \cos(\frac{1}{2}\tilde{\theta})\end{bmatrix} \simeq \begin{bmatrix}\frac{1}{2}\tilde{\boldsymbol{\theta}}\\ 1\end{bmatrix} \tag{38}$$

in which $\hat{\mathbf{k}}$ is the rotation axis and $\tilde{\theta}$ is the rotation angle. $\tilde{\boldsymbol{\theta}} = \hat{\mathbf{k}}\tilde{\theta}$ represents the rotation vector in $\mathrm{so}(3)$. Consequently, the residual equation of a corner feature is established as

$$\begin{bmatrix}\tilde{\boldsymbol{\theta}}_m\\ \tilde{\mathbf{p}}_m\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{T_q}\\ \mathbf{H}_{T_p}\end{bmatrix}\tilde{\mathbf{x}}_{T_m} + \begin{bmatrix}\mathbf{H}_{B_q}\\ \mathbf{H}_{B_p}\end{bmatrix}{}^{G}\tilde{\mathbf{B}} + \begin{bmatrix}\mathbf{n}_q\\ \mathbf{n}_p\end{bmatrix} \tag{39}$$

where $\tilde{\boldsymbol{\theta}}_m$ and $\tilde{\mathbf{p}}_m$ are the rotation and the position measurement residual of the corner, $\mathbf{H}_{T_q}$ and $\mathbf{H}_{T_p}$ are the measurement Jacobians with respect to the camera pose $\mathbf{x}_{T_m}$, $\mathbf{H}_{B_q}$ and $\mathbf{H}_{B_p}$ are the measurement Jacobians with respect to the corner feature ${}^{G}\mathbf{B}$, $\mathbf{n}_q \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_q^B)$ and $\mathbf{n}_p \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_p^B)$ are zero-mean Gaussian noises with rotation covariance $\mathbf{\Sigma}_q^B = \sigma_q\mathbf{I}_{3\times3}$ and position covariance $\mathbf{\Sigma}_p^B = \sigma_p\mathbf{I}_{3\times3}$. We empirically set $\sigma_q = 0.05$ and $\sigma_p = 0.01$ m in the implementation, based on the error level of the sensor used.

F. State Refinement and Layout Mapping

After the state update process with point features, high-level geometric features, including planes and box corners, are employed to further refine the system state.

Different from point features, all the planes and corners are regarded as SLAM features and initialized to the state vector as soon as there are sufficient observations, due to their extensive spatial presence and limited quantity. To improve computational efficiency, planes will be delayed marginalized from the state when they are lost for an extended period. If two planes have similar parameters and overlap, they will be merged into the first observed one. Moreover, to overcome the problem of accumulated errors during long-term navigation, we save the most meaningful planes and corners about the structure of the scene in the map. We argue that box corners are relatively static and stable regions, such as tables and wall corners. They are considered as layout components of environment configurations. Planes that are associated with box corners and with sufficiently large convex hull areas and ample observations are also considered reliable and dominant components. When their observations are lost, they are marked as layout landmarks instead of being marginalized. These corners and planes serve as permanent landmarks to provide layout-level long-term constraints.

Concretely, the state refinement is carried out in two stages: 1) layout alignment; and 2) joint refinement.

1) Layout Alignment: The layout alignment aims to get rid of the ghosting caused by drifts in the map and improve the global consistency of the state estimation and mapping. We solve this problem by applying the measurement function with fixed plane or corner landmarks in the EKF update to maintain their covariance. An observation of the box corner can provide full constraints to correct the current camera pose. When a new observation is associated with the layout landmark, a global transformation can be calculated to align the corners, and the camera pose is corrected to eliminate accumulated errors.

2) Joint Refinement: Subsequently, plane and corner landmarks as well as the camera pose are jointly refined through EKF update, using their respective measurement function described in the previous sections. The Chi-squared test is applied to reject outlier observations.

VI. Experiments

In this section, we comprehensively validate the overall performance and the essential techniques of the proposed CornerVINS. First of all, the camera localization accuracy and robustness are evaluated and compared against state-of-the-art methods using two public datasets. Subsequently, we assess four different variants of our system to show the strength of utilizing depth information and hierarchical features. We also provide a detailed discussion of the box corner, to demonstrate its importance and feasibility in practical scenarios. Finally, the computational cost is presented to showcase the efficiency of the system.

A. Test Benchmarks

To the best of our knowledge, there are few publicly available RGB-D inertial datasets for robots in structured indoor scenes, either lacking sufficient data sources or exhibiting a deficiency of planar structures. In this article, two datasets are used to demonstrate the effectiveness of our approach in practical applications.

1) CID-SIMS Dataset: The sequences in the CID-SIMS dataset [52] are captured in real-world indoor environments using a ground wheeled robot, which covers a wide array of real living and working scenes, including offices, open area workspaces, long corridors, stairs, and living apartments. This dataset provides ground truth trajectories for all the whole sequences, containing a variety of challenging motion modes (e.g., straight line motions and fast motions) as well as vision modes (e.g., motion blurs, dynamic objects, and weak textures). A total of 15 high-viewpoint sequences with adequate planes and corners are selected, which are suitable to measure various aspects of our approach.

2) VCU-RVI Dataset: Furthermore, the VCU-RVI dataset [53] offers several sequences for robots in a working building. However, ground truth trajectories are only available at the beginning and the end of each sequence, which can serve to assess the accumulated drift. We use four long sequences captured in a long corridor as the complement to enhance the results from the CID-SIMS dataset.

B. Experimental Setup

We make comparisons with a large variety of state-of-the-art systems, covering RGB-D methods, structure-based methods, and learning-based methods, using their open-source codes or executables, with suggested configurations.

1) VINS-Mono: A monocular visual-inertial navigation system utilizing point features based on the optimization framework [2].

2) OpenVINS: A filter-based system using monocular images and IMU data as input, where point features are divided into SLAM points and MSCKF points [3].

3) ORB-SLAM3: An optimization-based system using point features with capabilities for loop closing and global BA [4]. We use its RGB-D inertial mode for the experiments.

4) DroidSLAM: A deep learning-based SLAM system that iteratively updates the camera pose and the pixelwise depth. We adopt the RGB-D mode, where depth information serves as an additional penalty for the predicted depth [48].

5) DPVO: A deep monocular odometry exploiting the advantages of sparse image patches with a novel recurrent update operator and differentiable BA [49].

6) RTG-SLAM: A real-time RGB-D SLAM for large-scale environments using Gaussian splatting [50], whose back-end optimization is based on ORB-SLAM2 [21].

7) StructVIO: A structural line-based system using visual-inertial information [39], which is based on the AW assumption.

8) PlanarSLAM: An optimization-based RGB-D approach that extends ORB-SLAM2 [21] by exploiting points, lines, and planes on the assumption of the MW [38].

9) ManhattanSLAM: An RGB-D system that utilizes point, line, and plane features to estimate the camera pose, modeling the scene as the mixture of Manhattan frames [36].

10) ov_plane: A monocular VIO that extends OpenVINS by adding plane features and coplanar constraints [6].

TABLE I
RMSE ATE (↓) AND RMSE ARE (↓) OF THE COMPARED METHODS ON THE CID-SIMS DATASET

	Method	Office_1	Office_2	Office_3	Floor14_1	Floor14_3	14-13-12	Floor3_1	Floor3_2	Floor3_3	Floor13_1	Floor13_2	Apart1_2	Apart1_3	Apart2_2	Apart3_3
		42.01 m	90.31 m	95.25 m	103.70 m	180.43 m	21.89 m	85.61 m	150.55 m	196.46 m	130.43 m	135.10 m	77.18 m	154.00 m	85.88 m	147.96 m
ATE [m]	VINS-Mono [2]	/	0.158	/	0.412	/	/	/	0.525	2.073	/	2.655	/	0.355	0.146	0.346
	OpenVINS [3]	0.155	0.078	0.100	0.301	0.483	0.108	0.833	1.319	2.073	/	0.707	0.128	0.215	0.162	0.138
	ORB-SLAM3 [4]	0.102	0.106	0.061	0.261	0.417	2.824	0.280	4.936	0.325	0.897	0.469	21.118	0.692	0.630	2.706
	DroidSLAM [48]	0.099	0.222	0.215	0.223	0.472	**0.066**	0.308	0.537	1.073	0.733	0.330	0.281	0.461	0.151	0.189
	DPVO [49]	0.146	0.404	0.248	0.256	0.487	0.165	0.256	0.314	1.283	0.934	0.476	0.393	0.546	0.395	0.296
	RTG-SLAM [50]	0.058	0.057	0.046	/	1.575	1.216	0.429	0.687	0.915	/	/	/	/	/	/
	StructVIO [39]	0.254	0.278	0.348	1.067	8.480	0.081	1.501	3.355	6.822	3.809	2.991	0.348	0.449	0.275	0.664
	PlanarSLAM [38]	0.294	13.788	1.181	0.294	/	1.828	1.501	/	/	/	/	2.267	/	4.878	3.052
	ManhattanSLAM [36]	0.086	0.213	0.205	0.448	1.006	0.105	1.072	1.550	1.810	/	0.429	/	/	/	/
	ov_plane [6]	0.133	0.102	0.110	0.309	0.345	0.103	0.346	1.697	3.026	1.062	/	0.064	0.229	0.279	0.137
	StructurePLP [51]	0.049	1.702	/	/	/	2.397	1.315	/	19.641	12.314	/	1.152	0.979	/	/
	CornerVINS (Ours)	**0.022**	**0.030**	**0.041**	**0.075**	**0.172**	0.076	**0.107**	**0.287**	**0.294**	**0.078**	**0.192**	**0.037**	**0.054**	**0.038**	**0.044**
ARE [°]	VINS-Mono [2]	/	1.619	/	3.295	/	/	/	**1.149**	1.949	/	6.418	/	**1.921**	2.543	1.710
	OpenVINS [3]	1.184	1.616	**1.305**	1.310	3.475	1.617	1.645	2.265	2.757	/	2.266	**1.443**	3.753	2.280	1.423
	ORB-SLAM3 [4]	12.112	9.069	9.208	10.530	8.728	132.166	12.744	11.568	9.097	12.230	10.939	22.456	16.915	39.682	55.814
	DroidSLAM [48]	2.370	5.386	5.077	1.560	3.030	2.043	2.063	3.893	3.192	3.742	3.858	5.131	8.748	4.002	3.508
	DPVO [49]	2.374	4.176	5.468	1.393	2.799	3.746	**1.129**	1.535	2.334	2.472	2.107	3.980	6.595	4.524	3.126
	RTG-SLAM [50]	1.976	2.105	1.532	/	5.771	61.341	2.820	1.933	4.623	/	/	/	/	/	/
	StructVIO [39]	3.889	2.383	1.929	**0.876**	8.292	2.777	2.119	1.695	2.251	8.383	2.675	2.708	4.853	**1.523**	2.257
	PlanarSLAM [38]	7.505	58.886	34.266	2.172	/	40.274	5.706	/	/	/	/	68.405	/	32.964	67.915
	ManhattanSLAM [36]	1.239	5.594	3.322	3.926	12.709	3.370	8.137	6.242	4.548	/	2.447	/	/	/	/
	ov_plane [6]	1.584	2.732	1.834	1.091	2.110	2.358	1.154	2.997	3.222	1.967	/	1.759	1.988	2.909	1.841
	StructurePLP [51]	1.436	70.395	/	/	/	120.510	13.014	/	83.625	103.176	/	52.611	54.854	/	/
	CornerVINS (Ours)	**1.056**	**1.096**	1.393	1.303	**1.664**	**1.033**	1.226	1.175	**1.504**	**1.238**	**1.464**	1.754	1.958	1.628	**0.715**

* The best results for each sequence are boldfaced and the second best results are underlined.
* /indicates that the method fails in all five tests when the estimated trajectory is less than 50% complete or drifts larger than 50 m.

11) StructurePLP: An optimization-based approach incorporating the semantic and geometric features to boost both front-end pose tracking and back-end map optimization. We run the point-line-plane configuration for comparison. Because it requires instance planar segmentation masks as input, we use AHC to segment the planar regions offline, in the same manner as our system.

All the experiments are performed on an Intel i9-13900KS × 32 CPU at 3.20 GHz with 64 GiB of memory. DroidSLAM, DPVO, and RTG-SLAM are additionally powered by an NVIDIA GeForce RTX 4070 Ti GPU with 12 GiB of memory. We output online poses of all the algorithms for a fair comparison. The Umeyama algorithm [54] is used to align the estimated and the ground truth trajectories, without scaling except for monocular DPVO. The root mean square error (RMSE) of the absolute trajectory error (ATE) and the absolute rotation error (ARE) are the quantitative evaluation criteria [55]. Considering the randomness of pose estimation, we run each system five times and report the median results.

C. Overall Performance

We first conduct experiments to evaluate the localization accuracy of the proposed CornerVINS and compare it with the above baselines. As shown in Table I, CornerVINS achieves the lowest or the second lowest ATE and ARE on most sequences, surpassing prior work by large margins. The advantage is more obvious on long corridor sequences because rich data sources complement each other and the effective exploration of hierarchical features aids in understanding the environment at a high level. Other methods experience catastrophic failures or large drift on these challenging sequences.

VINS-Mono presents unsatisfactory performance because weak textures lead to fewer associated point features, causing the joint vision and inertia optimization to fail in reaching the correct convergence. Some sequences in the table show small ATEs and AREs. However, their trajectories are relatively incomplete. ORB-SLAM3 is more robust because of the assistance of depth, but it still tracks lost in weakly textured scenes. Although the relocalization module enables it to continue the navigation, the overall accuracy remains low on these sequences. Furthermore, pose jumps during the initialization phase is the main reason for large rotation errors. RTG-SLAM achieves tracking accuracy comparable with classical SLAM methods on some sequences, meanwhile estimating a large number of Gaussian parameters.

The optimization-based RGB-D systems (PlanarSLAM, ManhattanSLAM, and StructurePLP) adopt the ORB-SLAM frameworks by adding lines, planes, and structural information, and jointly optimize multiple features in the BA. With the help of high-level features, they are theoretically accurate when the test sequences have rich textures and low depth noise, as multiple features can complement each other in joint optimization. They work well on sequence Office_1 and sequence Office_2, which include sufficient rotational motion and loop closures. However, if one of the features encounters a fatal error, the entire system will collapse, making it vulnerable to noise and degraded environments, such as the floor sequences. In such cases, the potential of planes are not fully realized and many duplicate planes are generated by PlanarSLAM and ManhattanSLAM, because plane association fails under large drift. The point clouds generated by StructurePLP are globally inconsistent, which makes it difficult to extract planes and establish geometric errors. Without a good initial state, multivariate features and the assumption of structures are not delivering the expected value, especially in planar

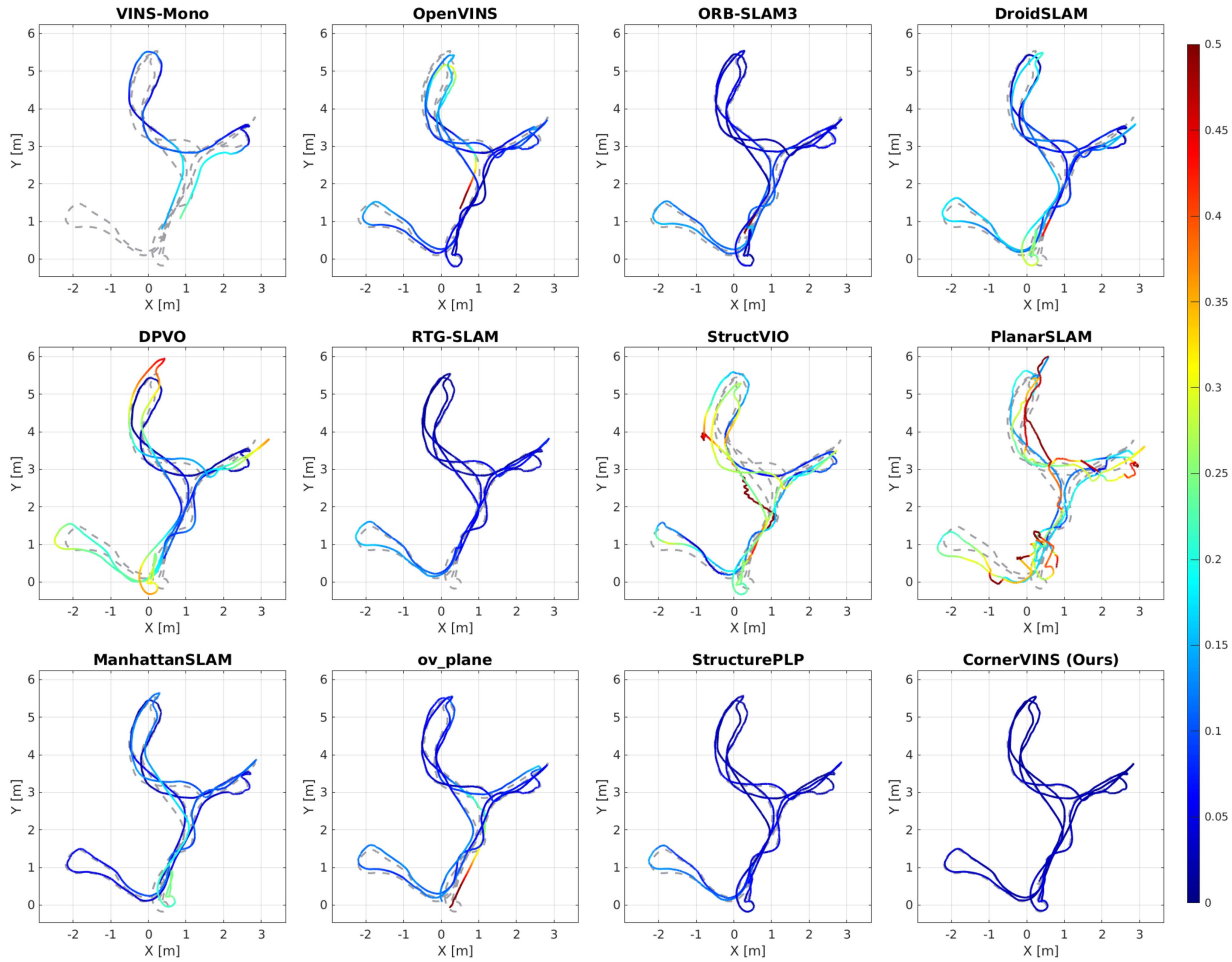

Fig. 8. Trajectories estimated by various methods on sequence Office_1 of the CID-SIMS dataset. The colored and dashed lines denote the estimated and ground truth trajectories, respectively. For visualization, poses from the first 60 seconds are used to align the estimated trajectories with ground truth trajectories. The color bar indicates the magnitude of the ATE.

ambiguous regions. In contrast, CornerVINS makes full use of the input information and can understand different scenarios at a high level. The state update and refinement processes adopt a hierarchical strategy, avoiding damage to the map, and the system errors are promptly suppressed in corners, providing a robust and superior guarantee for plane features.

The filter-based systems (OpenVINS, StructVIO, and ov_plane) work successfully on most sequences. However, due to the degradation of robot motion and large visual noise, the system state cannot be fully constrained, resulting in large cumulative errors. ov_plane outperforms OpenVINS on some sequences due to the additional point-on-plane constraints. However, the plane extraction accuracy is affected by insufficient points in weak texture regions and the occlusion of cluttered objects in real-world scenes, leading to limited improvements in many cases. StructVIO also suffers from noise and inaccurate initial motions in identifying the main directions of the AW.

The learning-based methods (DroidSLAM and DPVO) exhibit robust performance with reasonable accuracy on all sequences, without fine-tuning the provided models, showing the effectiveness of using deep network backbones. DPVO takes only monocular input, the performance is comparable on all sequences. DroidSLAM outperforms CornerVINS on a short sequence 14-13-12 in terms of ATE. This could be attributed to the movement pattern of this sequence (handheld collection) aligning more closely with the prior knowledge learned during training. Furthermore, as the robot continues to explore unknown areas, corners are not observed repeatedly, resulting in the absence of layout constraints. As a result, CornerVINS does not exhibit a substantial advantage on this sequence. The accuracy of DroidSLAM drops on other long sequences with degenerate motion. Another point worth noting is that these learning-based approaches seem less focused on rotational errors. For example, although DroidSLAM shows better ATE on sequence 14-13-12, the ARE is much larger than ours. On sequence Floor14_1 and Floor3_2, DroidSLAM has a significantly smaller translation error than StructVIO, while the rotation error is much larger. Nevertheless, they show great potential to be integrated with structural and geometric constraints into the depth estimation, to further increase the localization accuracy. Overall, DroidSLAM's performance is surpassed only by ours. We argue that the impressive performance is inseparable from their depth strategy, that is, taking depth as an optimization term to iteratively update the pixelwise depth, which is very meaningful for noisy

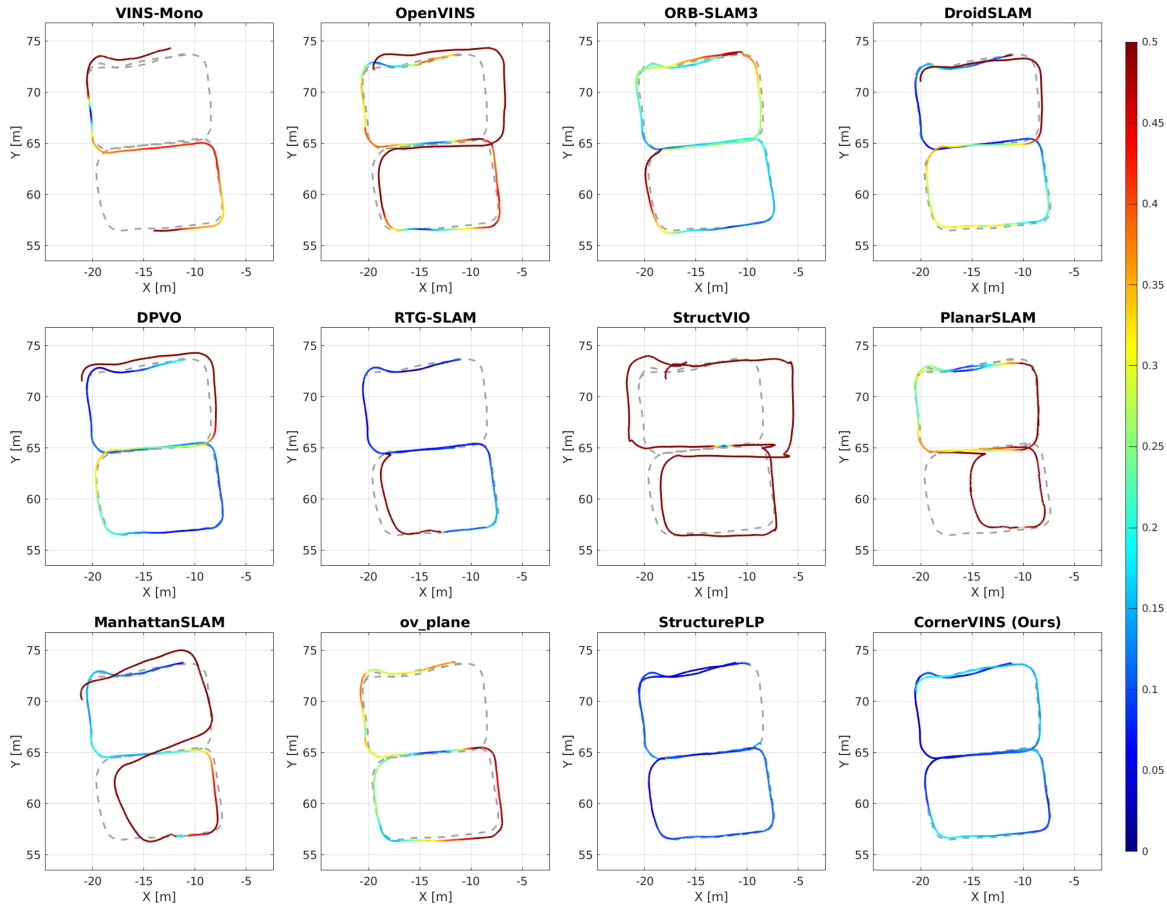

Fig. 9. Trajectories estimated by various methods on sequence Floor3_1 of the CID-SIMS dataset. The colored and dashed lines denote the estimated and ground truth trajectories, respectively. For visualization, poses from the first 60 seconds are used to align the estimated trajectories with ground truth trajectories. The color bar indicates the magnitude of the ATE.

environments. Our system takes into account the structural constraints and the intrinsic semantic information of the geometric elements in the environment, and can well identify noisy scenes and achieve accurate localization. Our approach is robust to a variety of challenging scenes and motion modalities with the proposed novelties, showing superior performance against drift. Figs. 8 and 9 display some examples of the estimated trajectories. Fig. 10 shows generated maps using our hierarchical features, where stable planes and corners make up layout maps.

The experimental results on the VCU-RVI dataset in Table II further confirm the effectiveness of the proposed method. In these sequences, the robot starts in a crowded lab, explores in it, then traverses a corridor, and finally returns to the lab. Because only the start and end trajectory errors are considered for evaluation on this dataset, we disable the loop closure modules of ORB-SLAM3, VINS-Mono, PlanarSLAM, and StructurePLP, and apply ATE to compare their accumulated errors, to better reflect the overall performance. As can be seen from Table II, our approach has the smallest ATE on sequences CM1 and CM2, and is comparable with DroidSLAM on sequences CB1 and CB2, using only sparse and structural information and free of training. Other methods bring in large ATEs after long-term navigation.

TABLE II
RMSE ATE [M] (↓) OF THE COMPARED METHODS ON THE VCU-RVI DATASET

Method	CM1	CM2	CB1	CB2
	206 m	210 m	206 m	210 m
VINS-Mono [2]	/	/	/	/
OpenVINS [3]	17.880	61.748	/	35.345
ORB-SLAM3 [4]	<u>0.356</u>	1.227	1.493	8.602
DroidSLAM [48]	0.725	<u>0.851</u>	**1.206**	**0.738**
DPVO [49]	/	2.960	6.533	2.525
RTG-SLAM [50]	/	/	/	/
StructVIO [39]	2.039	2.942	15.497	2.801
PlanarSLAM [38]	5.787	/	/	/
ManhattanSLAM [36]	8.119	/	5.864	/
ov_plane [6]	/	70.450	/	29.313
StructurePLP [51]	/	/	/	/
CornerVINS (Ours)	**0.248**	**0.494**	<u>1.231</u>	<u>0.772</u>

*The best results for each sequence are boldfaced and the second best results are underlined.
*/ indicates that the method fails in all five tests when the estimated trajectory is less than 90% complete or drifts larger than 100 m.

In addition, following [56], the completeness of the estimated trajectories, including the success rate (COMP) and the excellence rate (CMPL), are visualized in Fig. 11. CornerVINS does not drop frames even in visually challenging situations

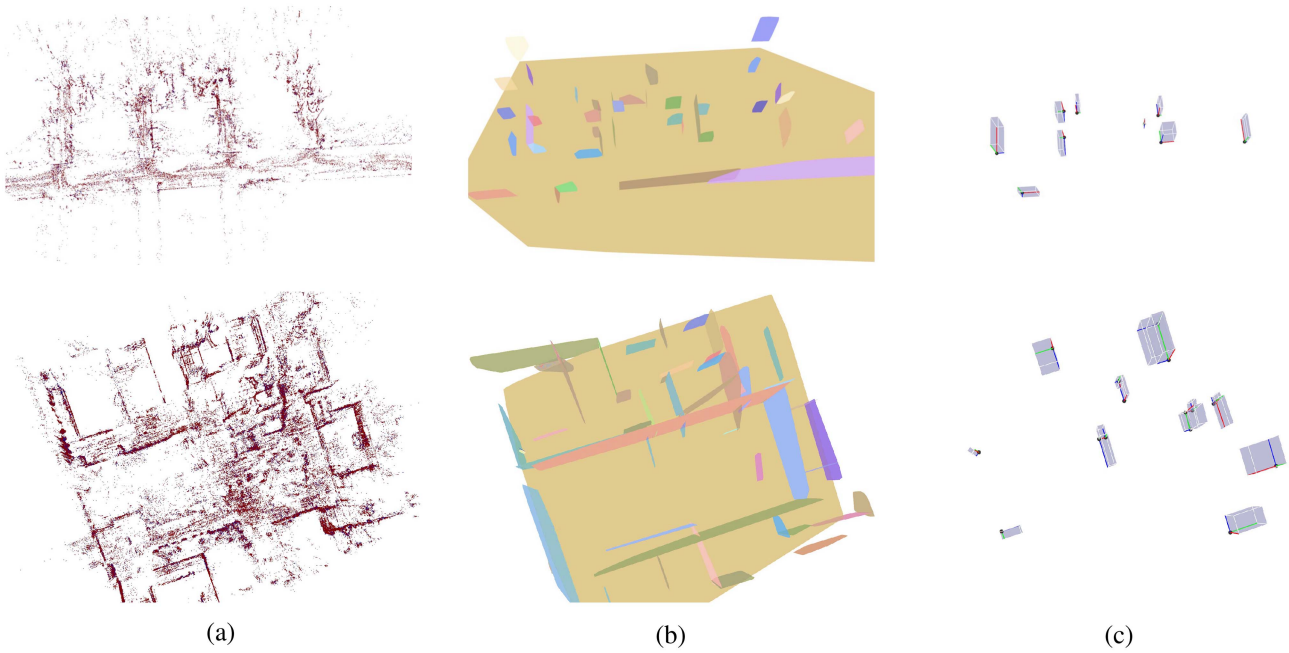

Fig. 10. Hierarchical maps constructed by CornerVINS. (a) Sparse Point Map. (b) Plane Map. (c) Corner Map. According to our state management strategy, only large and stable planes are saved in the final plane map while all corners that are observed multiple times are saved in the corner map because the number of corners is small.

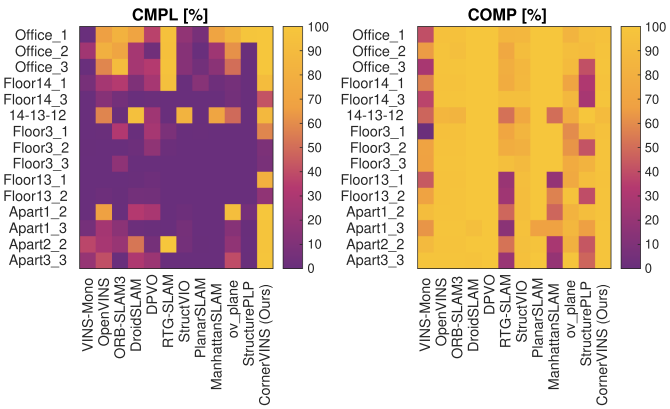

Fig. 11. Completeness of the evaluated trajectories by different methods on the CID-SIMS dataset. **CMPL** (↑) is defined by the ratio between the number of good poses (translation error <0.1 m) and the number of total frames while **COMP** (↑) means the ratio between the number of successful frames and the total number of frames.

and has high COMPs on all sequences. Though ORB-SLAM3 and OpenVINS have comparable COMPs, their CMPLs are low because of large drift. ov_plane is better than VINS-Mono in COMP. PlanarSLAM has a high degree of COMP on successful sequences but fails on other sequences. The learning-based DroidSLAM and DPVO achieve high COMP but lower CMPL compared with ours. Other methods, such as VINS-Mono and RTG-SLAM, generate many incomplete sequences and hence have poor completeness. In summary, CornerVINS exhibits superior performance in terms of accuracy and robustness, benefiting from the proposed innovative techniques.

TABLE III
DIFFERENT VARIANTS OF THE PROPOSED SYSTEM

Variant	Input Data			Geometric Feature		
	Color	Depth	IMU	Point	Plane	Corner
VINS-CIP (baseline)	✓		✓	✓		
VINS-CDIP (+depth)	✓	✓	✓	✓		
VINS-CDIPP (+plane)	✓	✓	✓	✓	✓	
CornerVINS (+corner)	✓	✓	✓	✓	✓	✓

D. Ablation Studies

We also perform ablation studies considering four variants of our system, as listed in Table III, to reveal the impact of fusing depth information and hierarchical features into the naive point-based VIO.

1) VINS-CIP: The RGB-inertial mode serving as a baseline.

2) VINS-CDIP: The RGB-D-inertial mode after introducing depth information into point triangulation (Section IV-C) and adding depth measurements (Section IV-D) based on VINS-CIP.

3) VINS-CDIPP: Planes are further added into the state vector and saved as permanent landmarks using a new marginalize strategy, as described in Section V-D and V-F.

4) CornerVINS: The full RGB-D-inertial version with hierarchical features including points, planes, and the proposed box corners.

The localization results are shown in Tables IV and V. As the absolute scale information is enhanced by exploiting depth information for point features, VINS-CDIP obtains a performance leap forward over its baseline VINS-CIP on all sequences. VINS-CDIPP works better than VINS-CDIP after adding plane

TABLE IV
RMSE ATE (↓) AND RMSE ARE (↓) OF DIFFERENT VARIANTS ON THE CID-SIMS DATASET

	Variant	Office_1	Office_2	Office_3	Floor14_1	Floor14_3	14-13-12	Floor3_1	Floor3_2	Floor3_3	Floor13_1	Floor13_2	Apart1_2	Apart1_3	Apart2_2	Apart3_3
ATE [m]	VINS-CIP	0.127	0.089	/	0.459	0.623	0.234	0.895	1.119	1.133	0.888	1.507	/	0.342	0.227	0.132
	VINS-CDIP	**0.032**	**0.071**	**0.083**	**0.105**	**0.214**	**0.084**	**0.206**	**0.586**	**0.449**	**0.191**	**0.188**	**0.060**	**0.082**	**0.048**	**0.101**
	VINS-CDIPP	**0.021**	**0.033**	**0.060**	**0.076**	**0.179**	0.088	**0.131**	**0.308**	**0.379**	**0.072**	0.217	**0.054**	**0.050**	**0.039**	**0.044**
	CornerVINS	0.022	**0.030**	**0.041**	**0.075**	**0.172**	**0.076**	**0.107**	**0.287**	**0.294**	0.078	**0.192**	**0.037**	0.054	**0.038**	**0.044**
ARE [°]	VINS-CIP	1.426	1.356	/	1.002	1.803	1.872	2.549	2.549	2.131	1.607	1.470	/	2.905	2.281	1.257
	VINS-CDIP	**1.229**	1.747	**1.31**	1.447	2.031	**1.198**	**1.531**	**2.448**	**1.783**	**1.487**	**1.405**	**1.908**	**1.763**	**1.671**	**1.080**
	VINS-CDIPP	**1.072**	**1.187**	1.321	**1.331**	**1.667**	**0.902**	**1.257**	**1.198**	**1.432**	**1.252**	**1.250**	1.921	1.792	**1.596**	**0.948**
	CornerVINS	**1.056**	**1.096**	1.393	**1.303**	**1.664**	1.033	**1.226**	**1.175**	1.504	**1.238**	1.464	**1.754**	1.958	1.628	**0.715**

* The above row is the baseline of the following row and the results that improve on top of the baseline are bolded.

TABLE V
RMSE ATE [M] (↓) OF DIFFERENT VARIANTS ON THE VCU-RVI DATASET

Variant	CM1	CM2	CB1	CB2
VINS-CIP	1.988	5.749	34.929	9.253
VINS-CDIP	**0.991**	**2.318**	**2.490**	**3.664**
VINS-CDIPP	**0.393**	**1.884**	**1.362**	**0.804**
CornerVINS	**0.248**	**0.494**	**1.231**	**0.772**

*The above row is the baseline of the following row and the results that improve on top of the baseline are bolded.

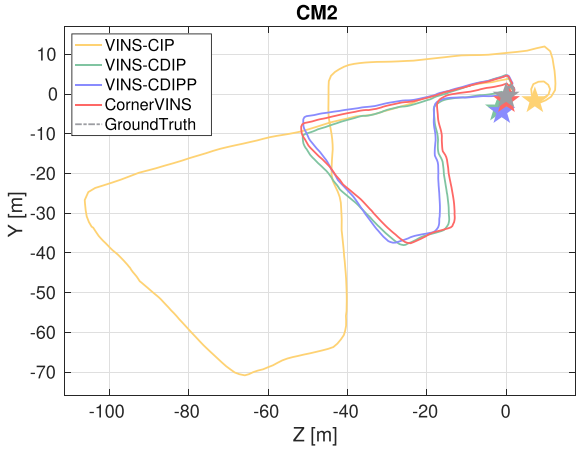

Fig. 12. Trajectories estimated by different variants of our system, on sequence CM2 of the VCU-RVI dataset. For visualization, poses from the first 60 s are used to align the estimated trajectories with ground truth trajectories. The stars denote the end points of the corresponding trajectories. CornerVINS achieves the lowest drift.

TABLE VI
STATISTICS OF BOX CORNERS ON THE CID-SIMS DATASET

Sequence	TF	DF	DC	Sequence	TF	DF	DC
Office_1	4250	38	2	Floor3_3	7299	68	8
Office_2	7396	24	3	Floor13_1	5336	24	9
Office_3	7132	40	3	Floor13_2	5820	31	9
Floor14_1	5530	177	11	Apart1_2	6179	19	3
Floor14_3	8918	200	20	Apart1_3	11280	23	8
14-13-12	2138	83	4	Apart2_2	10076	295	6
Floor3_1	3834	14	3	Apart3_3	10800	245	16
Floor3_2	5649	8	2	Avg.	6776	86	7.13

*TF means the total number of frames in this sequence, DF means the number of frames in which box corners are detected, and DC means the total number of box corners in the map.

TABLE VII
AVERAGE NUMBER OF PLANES ON THE CID-SIMS DATASET

Number of Layout Planes			Number of Total Planes		
w/o BC	**w/ BC**	GT	w/o BC	**w/ BC**	GT
44.47	**46.53**	53.00	216.27	**213.73**	190.47

*The number of layout planes means layout-level landmarks saved in the final map after merging and marginalization.
*The number of total planes means all landmarks created by the system.
*w/o BC = Without Box Corner, w/ BC = With Box Corner, GT = Ground Truth.

features into the system state. CornerVINS further improves the positioning accuracy depending on the introduction of high-level box corner features, which also promote the utilization of planes. On long sequences from the VCU-RVI dataset, no box corners are repeatedly observed to provide long-term relocalization constraints. Nevertheless, the performance improves due to the implicit plane structural constraints provided by the corners. Fig. 12 shows the trajectories generated by different variants. The proposed layout-level features can effectively suppress the cumulative error of the system, making CornerVINS more robust and accurate in long-term navigation.

E. Box Corner Features

We further verify the proposed box corner feature to prove its applicability and potential in practice. As visualized in Fig. 13, our approach correctly finds different types of box corners in these real-world scenes, such as table corners and wall corners. Based on the efficient plane contour representation, the detected corners are real corners in the scenes and their axial information is reliably identified. Box corners are commonplace for robots to provide high-level understandings and constraints in structured indoor environments. Table VI summarizes the statistics of corners on the CID-SIMS dataset. We count the number of frames where corners are observed (DF) and compare it with the total number of image frames (TF). The total number of corners (DC) for each sequence is also listed. The table reveals that an average of 7.13 box corners exist in these scenes and 1% frames observe a box corner on average. As demonstrated in Fig. 14, corners are scattered throughout the scenes with limited quantities and are well distinguished from each other by distance or direction. They act as markers of the scene layout and can be associated robustly. A small number of corner observations can effectively improve the overall accuracy.

Table VII shows the total number of planes and the final number of planes (layout planes) with (w/ BC) and without (w/o BC) box corner features. We construct a map in the same way but

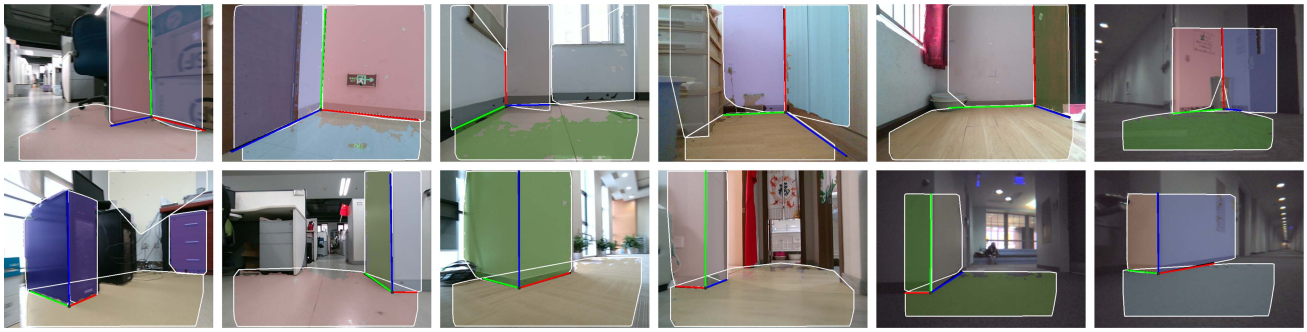

Fig. 13. Planes and box corners detected on the two datasets. The planes are randomly assigned colors and overlaid on the images, and the white outlines correspond to their convex hulls. The red, green, and blue lines are the intersection lines that form the box corners, representing the x, y, and z-axis of the corners in the camera coordinate system, respectively.

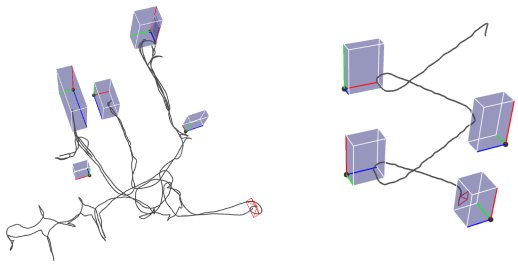

Fig. 14. Reconstructed corners and trajectories on the CID-SIMS dataset.

with ground truth trajectories and record the number of planes as references (GT). With box corner features, planes belonging to the same corner are first related using the hierarchical feature association, which is helpful under large drift. Our marginalization strategy saves planes with adequate observations and large areas as permanent landmarks to represent the scene layout. When data association is more accurate, more stable planes will be preserved in the final map, and the total number of generated planes will be smaller. It can be inferred from Table VII that the introduction of box corners effectively helps understand the scene and perform robust data association, so the number of planes (both total and final) goes closer to the GT. Fig. 15 intuitively shows comparisons of two final plane maps. By leveraging box corner features, planes are robustly associated and the consistency of the final map is greatly enhanced.

F. Runtime Analysis

The execution efficiency of the proposed system is tested on sequence Office_1 of the CID-SIMS dataset and compared with an RGB-D plane system called PlanarSLAM. Table VIII presents the average computational time required by each critical module of our system. The point detection, tracking, and update modules run in parallel with the plane and box corner extraction modules. Each feature update module includes the EKF update process and the corresponding map update process, such as marginalization and plane merging. It can be inferred that the main execution time is spent on plane extraction, while corner extraction and update take very little time consumption, as the number of corners is limited. Furthermore, Fig. 16 provides a comparison with PlanarSLAM. Due to the new state management and data association strategy, the running time of our algorithm grows slowly. In contrast, PlanarSLAM runs slower when the number of planes increases. Besides, our system is free of BA optimization, so the overall time is superior to PlanarSLAM. Consequently, the proposed method significantly improves the performance of structured indoor navigation while introducing little computational cost.

TABLE VIII
AVERAGE COMPUTATIONAL TIME PER FRAME

Module		Time [ms]
IMU Processing	IMU Integration Propagation and Clone	0.182
Feature Extraction	Point Detection Plane Extraction Corner Extraction	0.512 15.173 0.010
Feature Association	Point Tracking Plane & Corner Association	0.839 0.061
State Management	Point Feature Update Plane Feature Update Corner Feature Update	7.610 5.897 0.001
Total		22.168

VII. DISCUSSION

At last, we discuss the limitations and prospects of the proposed system, hoping this will provide helpful insights to the community.

A. Dependency on Planar Information

Despite the superior performance of the box corner in both accuracy and efficiency, the proposed method relies on stable layout structures, primarily large planes. As a result, the performance may degrade when the scene is dynamic or overwhelmingly cluttered. Moreover, in environments with sparse structures, CornerVINS becomes an RGB-D point system.

B. Improved Strategies for Corner Orientation

Since the detected planes making up a corner are not completely vertical, we employ SVD decomposition for approximation, which inherently contains errors. It would be beneficial to

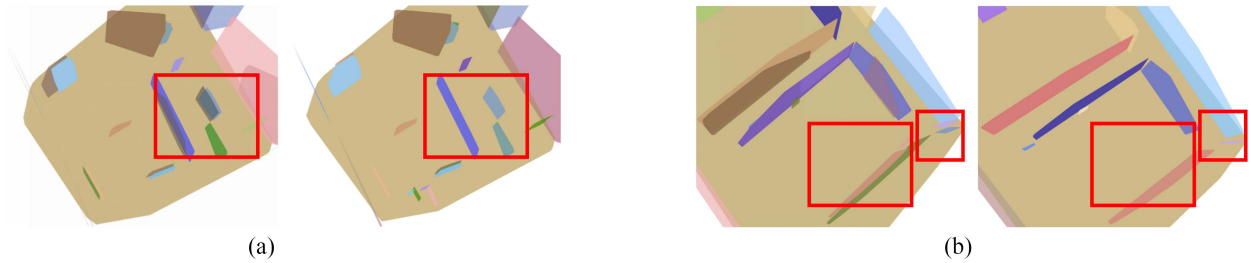

Fig. 15. Comparison of final plane maps without box corners (left) and with box corners (right). The colors are randomly assigned for each run.

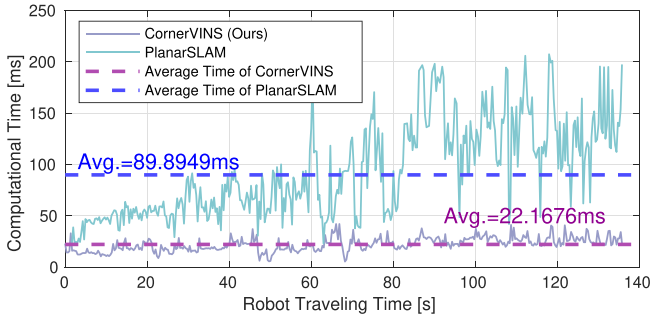

Fig. 16. Runtime comparison on each frame.

assign different convergence matrices depending on how close the box edges are to being orthogonal. Moreover, due to the size and view limitation, it is virtually impossible for robots to observe corners with similar orientations at close distances. Therefore, the angle threshold is manually selected in the association after comparing adaptive position conditions. Although experiments have demonstrated that this methodology ensures the distinguishability of corners, developing a more intelligent strategy for box corner orientations could be a promising direction, to deal with complex scenes and for the deployment on various types of robotic devices. For instance, in order to enhance the accuracy of corner direction extraction, we anticipate that a feasible solution is to establish constraints between line segments extracted from color images and intersection lines derived from depth.

C. Integration With Structural Assumption

The box corner takes advantage of local structural information without making any prior assumptions about the scene. By considering position information, it can cope with noisy environments with imperfect structural situations. Despite this, the box corner can adaptively integrate with structural assumptions, such as the MW assumption, by allowing multiple corners to share rotation components or specific directions, to enhance its performance in well-structured scenes.

D. Practical Applications

To further assess the performance in real-world applications, we test our algorithm on an embedded device, which is equipped with a 4-core processor at 2.20 GHz and 3.7 GiB of memory. Our system demonstrates the ability to operate on devices with low-frequency processors and limited memory while maintaining comparable accuracy. However, it is approximately 3.5 times slower than on the workstation. More efficient plane extraction algorithms, intelligent parallelization designs, or using GPU for acceleration could be beneficial. Furthermore, downsampling input images and point clouds during plane fitting will significantly improve efficiency on resource-constrained devices.

VIII. Conclusion

This article presents CornerVINS, a novel RGB-D inertial navigation system that utilizes hierarchical geometric features for robot localization and mapping in structured environments. We enhance point features in the EKF framework by integrating depth information during the triangulation and state update process. Planes are modeled as bounded regions using convex hulls to increase their uniqueness in data association and further refine the system state. In addition, we introduce box corner features, which enable robust data association in planar ambiguous regions and provide full constraints for motions, to sufficiently utilize structural information and understand the environments at a high level. Box corners and stable planes are considered layout components of scenes and serve as permanent landmarks to correct camera poses. We conduct experiments in real-world navigation circumstances and demonstrate that CornerVINS significantly outperforms existing methods in terms of localization accuracy, robustness, and efficiency. The core implementations and parameter configurations are in https://github.com/zydddd/CornerVINS.

The proposed box corners bridge geometric and semantic dimensions for understanding the environment layout at a high level, highlighting the potential for advanced robotics applications in real-world scenarios. In the future, we plan to explore the scene structure at different levels to take a further step toward accurate, robust, and efficient robotic intelligence.

References

[1] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 3565–3572.
[2] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[3] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4666–4672.

[4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[5] L. Lipson, Z. Teed, and J. Deng, "Deep patch visual SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2025, pp. 424–440.

[6] C. Chen, P. Geneva, Y. Peng, W. Lee, and G. Huang, "Monocular visual-inertial odometry with planar regularities," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 6224–6231.

[7] Y. Wu et al., "An object SLAM framework for association, mapping, and high-level tasks," *IEEE Trans. Robot.*, vol. 39, no. 4, pp. 2912–2932, Aug. 2023.

[8] F. Tang, Y. Wu, X. Hou, and H. Ling, "3D mapping and 6D pose computation for real time augmented reality on cylindrical objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 2887–2899, Sep. 2020.

[9] K. Joo, P. Kim, M. Hebert, I. S. Kweon, and H. J. Kim, "Linear RGB-D SLAM for structured environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8403–8419, Nov. 2022.

[10] K. Ram, C. Kharyal, S. S. Harithas, and K. M. Krishna, "RP-VIO: Robust plane-based visual-inertial odometry for dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 9198–9205.

[11] C. Arndt, R. Sabzevari, and J. Civera, "Do planar constraints improve camera pose estimation in monocular SLAM?," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 2213–2222.

[12] Z. Shan, R. Li, and S. Schwertfeger, "RGBD-inertial trajectory estimation and mapping for ground robots," *Sensors*, vol. 19, no. 10, 2019, Art. no. 2251.

[13] Y. Yang, P. Geneva, X. Zuo, K. Eckenhoff, Y. Liu, and G. Huang, "Tightly-coupled aided inertial navigation with point and plane features," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6094–6100.

[14] X. Yang, Z. Yuan, D. Zhu, C. Chi, K. Li, and C. Liao, "Robust and efficient RGB-D SLAM in dynamic environments," *IEEE Trans. Multimedia*, vol. 23, pp. 4208–4219, 2021.

[15] A. Harati and R. Siegwart, "Orthogonal 3D-SLAM for indoor environments using right angle corners," in *Proc. 3rd Eur. Conf. Mobile Robots*, 2007, pp. 144–149.

[16] X. Hu, Y. Wu, M. Zhao, L. Yang, X. Zhang, and X. Ji, "PAS-SLAM: A visual SLAM system for planar ambiguous scenes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 3, pp. 2026–2044, Mar. 2025.

[17] S. Yang, Y. Song, M. Kaess, and S. Scherer, "Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1222–1229.

[18] D. Chen et al., "VIP-SLAM: An efficient tightly-coupled RGB-D visual inertial planar SLAM," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 5615–5621.

[19] J. Jiang, J. Wang, P. Wang, P. Bao, and Z. Chen, "LiPMatch: LiDAR point cloud plane based loop-closure," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6861–6868, Oct. 2020.

[20] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9107–9114, Oct. 2022.

[21] R. Mur-Artal and J. D. Tardós, "Orb-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[22] T. Schops, T. Sattler, and M. Pollefeys, "Bad SLAM: Bundle adjusted direct RGB-D SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 134–144.

[23] J. Yuan, S. Zhu, K. Tang, and Q. Sun, "ORB-TEDM: An RGB-D SLAM approach fusing ORB triangulation estimates and depth measurements," *IEEE Trans. Instrum. Meas.*, vol. 71, 2022, Art. no. 5006315.

[24] P. Gu and Z. Meng, "S-VIO: Exploiting structural constraints for RGB-D visual inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3542–3549, Jun. 2023.

[25] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas, "Tracking an RGB-D camera using points and planes," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2013, pp. 51–58.

[26] C. X. Guo and S. I. Roumeliotis, "IMU-RGBD camera navigation using point and plane features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3164–3171.

[27] M. Hsiao, E. Westman, and M. Kaess, "Dense planar-inertial SLAM with structural constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*. IEEE, 2018, pp. 6521–6528.

[28] H. M. Cho, H. Jo, and E. Kim, "SP-SLAM: Surfel-point simultaneous localization and mapping," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 5, pp. 2568–2579, Oct. 2022.

[29] X. Li, Y. Li, E. P. Örnek, J. Lin, and F. Tombari, "Co-planar parametrization for stereo-SLAM and visual-inertial odometry," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6972–6979, Oct. 2020.

[30] H. Wang et al., "RSS: Robust stereo SLAM with novel extraction and full exploitation of plane features," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 5158–5165, Jun. 2024.

[31] P. Kim, B. Coltin, and H. J. Kim, "Linear RGB-D SLAM for planar environments," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 333–348.

[32] K. Joo, T.-H. Oh, F. Rameau, J.-C. Bazin, and I. S. Kweon, "Linear RGB-D SLAM for Atlanta world," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1077–1083.

[33] J. Liu and Z. Meng, "Visual SLAM with drift-free rotation estimation in manhattan world," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6512–6519, Oct. 2020.

[34] P. Kim, B. Coltin, and H. J. Kim, "Low-drift visual odometry in structured environments by decoupling rotational and translational motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7247–7253.

[35] H. Li et al., "Hong Kong world: Leveraging structural regularity for line-based SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13035–13053, Nov. 2023.

[36] R. Yunus, Y. Li, and F. Tombari, "ManhattanSLAM: Robust planar tracking and mapping leveraging mixture of Manhattan frames," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 6687–6693.

[37] Y. Li, N. Brasch, Y. Wang, N. Navab, and F. Tombari, "Structure-SLAM: Low-drift monocular SLAM in indoor environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6583–6590, Oct. 2020.

[38] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari, "RGB-D SLAM with structural regularities," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11581–11587.

[39] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, "StructVIO: Visual-inertial odometry with structural regularity of man-made environments," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 999–1013, Aug. 2019.

[40] E. Fernández-Moral, P. Rives, V. Arévalo, and J. González-Jiménez, "Scene structure registration for localization and mapping," *Robot. Auton. Syst.*, vol. 75, pp. 649–660, 2016.

[41] Z. Deng, Y. Zhang, Y. Wu, Z. Ge, X. Hu, and W. Sun, "Object-plane co-represented and graph propagation-based semantic descriptor for relocalization," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11023–11030, Oct. 2022.

[42] M. Shaheer, J. A. Millan-Romera, H. Bavle, J. L. Sanchez-Lopez, J. Civera, and H. Voos, "Graph-based global robot localization informing situational graphs with architectural graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 9155–9162.

[43] C. Sommer, Y. Sun, L. Guibas, D. Cremers, and T. Birdal, "From planes to corners: Multi-purpose primitive detection in unorganized 3D point clouds," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1764–1771, Apr. 2020.

[44] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 430–443.

[45] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, vol. 2, pp. 674–679.

[46] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6218–6225.

[47] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: LiDAR-inertial 3D plane SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 123–130.

[48] Z. Teed and J. Deng, "Droid-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 16558–16569.

[49] Z. Teed, L. Lipson, and J. Deng, "Deep patch visual odometry," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 39033–39051..

[50] Z. Peng et al., "RTG-SLAM: Real-time 3D reconstruction at scale using Gaussian splatting," in *Proc. ACM SIGGRAPH Conf. Papers*, 2024, pp. 1–11.

[51] F. Shu, J. Wang, A. Pagani, and D. Stricker, "Structure PLP-SLAM: Efficient sparse mapping and localization using point, line and plane for monocular, RGB-D and stereo cameras," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 2105–2112.

[52] Y. Zhang et al., "CID-SIMS: Complex indoor dataset with semantic information and multi-sensor data from a ground wheeled robot viewpoint," *Int. J. Robot. Res.*, vol. 43, no. 7, pp. 899–917, 2024.

[53] H. Zhang, L. Jin, and C. Ye, "The VCU-RVI benchmark: Evaluating visual inertial odometry for indoor navigation applications with an RGB-D camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6209–6214.
[54] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.
[55] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.
[56] L. Jinyu, Y. Bangbang, C. Danpeng, W. Nan, Z. Guofeng, and B. Hujun, "Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality," *Virtual Reality Intell. Hardware*, vol. 1, no. 4, pp. 386–410, 2019.

Yidi Zhang received the B.E. degree in software engineering from Chongqing University, Chongqing, China, in 2021. She is currently working toward the Ph.D. degree in computer applied technology under the supervision of Prof. Yihong Wu with School of Artificial Intelligence, University of Chinese Academy of Sciences, and with State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

Her research interests include visual simultaneous localization and mapping, sensor fusion and real-time localization.

Fulin Tang received the Ph.D. degree in computer vision from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2020.

He is currently an Associate Professor with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include visual simultaneous localization and mapping, multi-sensors fusion for real-time localization and mapping and augmented reality applications.

Yihong Wu received the Ph.D. degree in applied mathematics from Mathematics Mechanization of Research Center, Institute of Systems Science, Chinese Academy of Sciences, Beijing, China, in 2001.

She is currently a Professor at State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences. Her research interests include image matching, camera calibration, camera pose determination, 3-D reconstruction, and SLAM.