Depth-Visual-Inertial (DVI) Mapping System for Robust Indoor 3D Reconstruction

Charles Hamesse[1,2], Michiel Vlaminck[2], Hiep Luong[2], and Rob Haelterman[1]

***Abstract*—We propose the Depth-Visual-Inertial (DVI) mapping system: a robust multi-sensor fusion framework for dense 3D mapping using time-of-flight cameras equipped with RGB and IMU sensors. Inspired by recent developments in real-time LiDAR-based odometry and mapping, our system uses an error-state iterative Kalman filter for state estimation: it processes the inertial sensor's data for state propagation, followed by a state update first using visual-inertial odometry, then depth-based odometry. This sensor fusion scheme makes our system robust to degenerate scenarios (e.g. lack of visual or geometrical features, fast rotations) and to noisy sensor data, like those that can be obtained with off-the-shelf time-of-flight DVI sensors. For evaluation, we propose the new Bunker DVI Dataset, featuring data from multiple DVI sensors recorded in challenging conditions reflecting search-and-rescue operations. We show the superior robustness and precision of our method against previous work. Following the open science principle, we make both our source code and dataset publicly available.**

***Index Terms*—Mapping, Localization, RGB-D Perception, Search and Rescue Robots**

I. Introduction

In the context of defence and security operations, search and rescue missions or emergency response, accurate and up-to-date 3D maps are critical, demanding lightweight and robust sensor systems capable of rapid deployment in challenging environments. Traditional LiDAR-based mapping systems, often mounted on tripods or mobile platforms, offer high accuracy but their deployment and operation remains difficult in these resource- and time-constrained scenarios. Recent advancements in solid-state LiDARs have enabled smaller, more portable systems [1]. However, even these may be unsuitable for highly dynamic environments or scenarios requiring minimal user burden, i.e. operating in a hands-free manner. Depth cameras present various advantages as they are lightweight, small enough to be worn or helmet-mounted, less expensive and, finally, consume less power. However, they suffer from severe accuracy and range limitations compared to LiDARs. Recent developments in 3D mapping with specific adaptations for depth sensors such as SSL-SLAM [2] or VoxelMap [3] only use depth maps, even though off-the-shelf depth cameras often come with hardware-synchronized, integrated RGB camera and IMU sensor [4], [5]. This makes them sensitive to fast motion and degenerate environments lacking abundant geometrical features. To address these challenges, we propose the depth-visual-inertial (DVI) mapping system, a robust multi-sensor fusion framework for dense 3D mapping using lightweight DVI sensors. On top of the depth-based odometry and mapping error-state iterative Kalman filter (ESIKF) proposed in VoxelMap [3], we integrate IMU-based state propagation, visual-inertial odometry state correction using a loosely-coupled formulation. The system is illustrated in Figure 1. We make our system more robust to visual and geometrical degeneracies by implementing a novel adaptative covariance estimation method as well as a solution remapping algorithm, extending the method proposed initially in [6]. To evaluate the performance of our system, we collect a new dataset with three DVI sensors (based on passive stereo, active stereo and time-of-flight). Our results show that our method is a viable option for real-life mapping scenarios in complex indoor environments. We open source our code[1] and dataset[2].

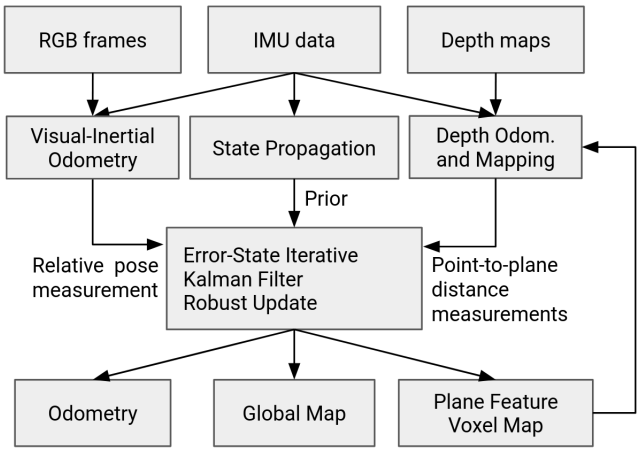

Fig. 1. Block diagram illustrating the full pipeline of the proposed depth-visual-inertial (DVI) mapping system.

Our contributions are the following:

- The Bunker DVI Dataset: a new dataset captured with three different DVI sensors (passive stereo, active stereo and time-of-flight) in a bunker-like environment with challenging trajectories to reflect the conditions of search-and-rescue operations.

Manuscript received: July 22, 2024; Revised September 27, 2024; Accepted October 13, 2024.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This work was funded in part by Belgium's Royal Higher Institute for Defence (RHID) under Grant DAP18/04 and Grant DAP22/01.

[1]C. Hamesse and R. Haelterman are with the Royal Military Academy, Department of Mathematics, Brussels, Belgium

[2]C. Hamesse, M. Vlaminck and H. Luong are with Ghent University, imec - IPI - URC, Ghent, Belgium

E-mail: charles.hamesse@mil.be

Digital Object Identifier (DOI): see top of this page.

[1]Code: https://github.com/charleshamesse/dvi-mapping-system

[2]Dataset: https://charleshamesse.github.io/bunker-dvi-dataset/

- The DVI Mapping System: A sensor fusion system based on the error-state iterative Kalman filter for DVI mapping, including two novel components:
  - An adaptative covariance estimation method to fuse visual-inertial odometry observations in the ESIKF.
  - A soft solution remapping method with auto-threshold to handle degenerate observations.
- Novel insights on the performance of DVI sensors and methods for 3D reconstruction based on the extensive evaluations of our algorithm and its ablated variants, compared against state-of-the-art and industry standard methods.

II. Related work

We review recent DVI sensors as well as LiDAR- and visual-inertial odometry and mapping algorithms.

A. DVI sensors

Most common off-the-shelf DVI sensors are marketed as depth cameras, but come in the form of small, lightweight systems featuring a depth sensor, an RGB camera and an IMU. Recent depth cameras mainly use active and passive stereo as well as time-of-flight (ToF) sensors to estimate depth. As shown in Table I, these sensors have interesting characteristics in terms of portability (size, weight and power) but a limited operational range. For mapping applications in indoor spaces, active depth sensing is preferred since these techniques are less affected by a lack of texture (e.g. with flat walls) [7]. They also require less computational power to reconstruct the depth maps compared to stereo matching-based devices.

B. DVI odometry and mapping

We propose to categorize the methods for odometry and mapping with DVI sensor inputs in three groups, depending on the way the depth information is processed:

- RGB-D: The depth map, combined with the RGB image, allows for a direct estimation of 3D points, simplifying the feature extraction, matching and triangulation process. Popular RGB-D-based VIO algorithms include ORB-SLAM3 [8] or VINS-Fusion (a variant of VINS-Mono [9] with RGB-D support) or RTAB-Map [10], commonly used in the Robot Operating System (ROS) community. There are also more recent developments such as VINS-RGB-D-FAST [11], which builds upon VINS-Fusion and brings a suite of refinements to increase robustness and accelerate the system's performance in resource-constrained embedded platform.
- Stereo: Two images from slightly different, rigidly attached viewpoints enable the estimation of depth through triangulation. Then, common stereo-based VIO systems employ feature matching for spatial stereo (using left and right images) and temporal stereo (using the history of left and right images). Examples of stereo-based VIO algorithms include VINS-Fusion (in its stereo variant), OpenVINS [12] or Basalt [13]. All of these systems are tightly-coupled systems using either sliding window optimization or iterative Kalman filtering. In [13], [14], B-spline continuous trajectory representations are used in order to perform more precise sensor fusion, at the cost of some computational expense.
- Point cloud-based: These methods operate on a point cloud representation of the depth map obtained from the sensor and employ variants of the iterative closest point (ICP) algorithm to register successive point clouds and estimate the relative pose between them. As such, these methods borrow heavily from the field of LiDAR odometry and mapping, where innovations in the last few years are extremely abundant [15]–[22]. However, we fail to see developments of similar maturity in the case of depth cameras. In fact, state-of-the-art LiDAR-based 3D mapping methods such as FAST-LIO [21], [22], RxLive [23], [24] or LVI-SAM [17] have not been successfully applied to depth cameras. This can be attributed to the noisier and scarcer nature of the depth maps compared to LiDAR scans, due to inherent sensor limitations. The most recent methods for dense 3D mapping with depth cameras include SSL-SLAM [2] or VoxelMap [3], both of which report results with the Intel Realsense L515 and the Microsoft Kinect for Azure. SSL-SLAM implements a scan-to-map registration method operating on edge and planar features to increase the tracking performance. VoxelMap implements scan-to-map registration with an ESIKF and a fully probabilistic voxel octree, where each voxel contains the parameters of a plane feature, including covariance updates arising from both point measurement noise and pose estimation error.

Conceptually, our system shares similarities with R2Live [23] or R3Live [24]. They are visual extensions of a LiDAR-inertial method, our work is a visual-inertial extension of a depth camera-based method. Then, they rely on Perspective-n-Point reprojection for visual features, while we use the full VIO capacity from VINS-Mono (including its rolling shutter compensation component to maintain usability with common off-the-shelf sensors) and implement a solution remapping algorithm to improve the robustness.

C. DVI datasets

Common datasets for indoor 3D reconstruction with depth sensors include TUM RGB-D [25], ICL-NUIM [26], NYU Depth V2 [27], or OpenLORIS [28]. However, none of these datasets combines i) multiple DVI sensors, needed to perform a thorough system-wise comparison, ii) challenging sequences with rapid motion and degenerate scenes in both the visual (e.g. texture-less flat walls) and the geometrical domain (pointing at a single planar structure such as the ground or a wall), and iii) the scale necessary to represent a search-and-rescue operation context in a large-scale area. To the best of our knowledge, such a dataset does not exist in the public domain. For this reason, we have collected such a dataset and present it in the next section.

III. Bunker DVI Dataset

We introduce the Bunker DVI Dataset, featuring three DVI sensors with different depth sensing modalities (passive stereo,

TABLE I
MAIN SPECIFICATIONS OF COMMON LIGHTWEIGHT, LOW-POWER AND SMALL OFF-THE-SHELF DEPTH-VISUAL-INERTIAL SENSORS. THE RANGE COLUMN INDICATES THE OPERATIONAL RANGE GIVEN BY THE MANUFACTURER.

Product	**Depth sensing technology**	**RGB shutter**	**Range [m]**	**Size [mm]**	**Weight [g]**	**Power [W]**
Realsense D455	Active stereo	Global	0.60 - 6.00	124 × 26 × 36	390	3.5
Realsense L515	MEMS LiDAR	Rolling	0.25 - 9.00	61 × 61 × 26	100	3.5
Kinect for Azure	Time-of-Flight	Rolling	0.25 - 3.00	103 × 39 × 126	440	5.9
Zed Mini	Passive stereo	Rolling	0.15 - 24.00	124 × 30 × 26	63	1.9

Fig. 2. Multi-sensor rig used to collect the DVI dataset. From top to bottom: Livox Mid-360, Sevensense Core Research, Intel Realsense D455F, Microsoft Kinect for Azure.

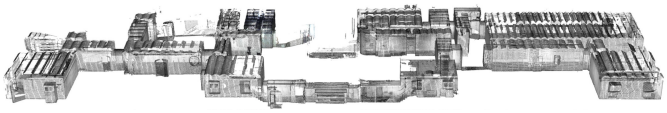

Fig. 3. 3D view of the Bunker DVI Dataset environment, extracted from the ground truth map.

active stereo and time-of-flight) and designed to simulate time-constrained mapping operations in adverse conditions, such as those in the context of search-and-rescue missions. The dataset has the following features:

- Different depth sensing modalities: our 3D-printed hand-held setup, shown in Figure 2, incorporates the Sevensense Core Research (passive stereo), Intel Realsense D455F (active stereo), and Microsoft Kinect for Azure (time-of-flight) sensors[3].
- Hardware and software synchronization: The Sevensense Core Research sensor and Livox Mid-360 are synchronized using hardware PTP, while the other cameras (USB-connected) rely on software synchronization.
- Challenging environment: the dataset was captured in a large, 1600m^2 bunker-like subterranean environment made of rooms of all shapes and sizes, including corridors, rooms, and various objects, mirroring typical search-and-rescue environments. Figure 3 depicts the environment.
- Challenging data sequences: our dataset includes regular data sequences as well as more complex sequences with aggressive motion, dynamic human activity within the field of view, degenerate scenarios (both geometrical and visual) as well as abrupt lighting changes. An overview of the sequences included in the dataset is given in Table II and illustrated in Figure 4.

[3]This sensor is now marketed as Femto Bolt [29]

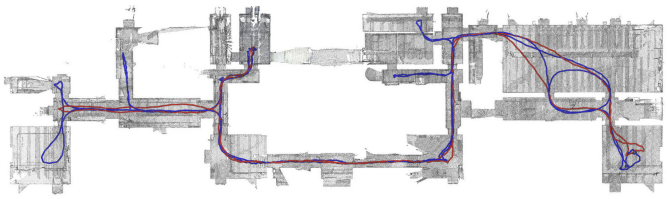

Fig. 4. Top-down view of two trajectories (Reg-1 and Reg-2) out of the six evaluation sequences in the Bunker DVI Dataset.

TABLE II
LIST OF SEQUENCES IN THE BUNKER DVI DATASET

Name	Description	Length [m]	Duration [s]
Calibration sequences			
Static	Different static poses	0.0	65
Dynamic	Fast rotation and translation	0.0	33
Warm-up sequences			
Small	Small room loop	0.0	45
Large	Large room loop	0.0	53
Evaluation sequences			
Reg-1	Smooth motion, stable features	210	347
Reg-2	Faster motion, narrower spaces	297	551
Deg-Vis-1	Person moving in FoV	234	328
Deg-Vis-2	Person moving, unstable lighting	250	410
Deg-Geo-1	Light geometrical degeneracy	159	203
Deg-Geo-2	Strong geometrical degeneracy	74	85

A survey-grade ground truth map collected using a FARO Focus laser scanner is provided. Similarly to recent SLAM datasets such as MCD-VIRAL [30] or the Newer College Dataset [31], the ground truth trajectory is determined by registering the Livox Mid-360 point clouds to the ground truth map after de-skewing. Extrinsic calibration was then used to compute the trajectories of each depth camera. For advanced users, we also provide SE3 B-spline trajectories computed using the Ceva library[4], offered with the MCD-VIRAL Dataset.

IV. DVI MAPPING SYSTEM

Our system performs state estimation by fusing the data from IMU, Visual-Inertial Odometry (VIO) and Depth Odometry and Mapping (DOM) in the ESIKF. Afterwards, the map is built by reprojecting the point clouds using the estimated poses. The main components are shown in Figure 1.

[4]https://github.com/mcdviral/ceva

A. Notations and conventions

Matrices and column vectors are written as upper- and lower-case bold symbols respectively. $\mathbb{R}^n$ denotes the n-dimensional real vector space. $(\cdot)^T$ denotes the transpose of a matrix or vector. Rotation and transformation matrices from one reference frame to another are denoted with a left superscript and a right subscript in the form: e.g., ${}^A\mathbf{T}_B$ is the transformation matrix from frame B to frame A. Similarly, a left superscript added to a vector denotes the frame in which it is expressed: ${}^A\mathbf{x}$ is expressed with respect to reference frame A. The Euclidean norm is noted $\|\mathbf{x}\|$, the Mahalanobis norm is noted and defined as $\|\mathbf{x}\|_{\mathbf{\Sigma}} = \sqrt{\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}}$. The skew-symmetric matrix of a vector $\mathbf{v} \in \mathbb{R}^3$ is noted $[\mathbf{v}]_\times$. We make use of the Lie groups SO(3) and SE(3), the manifold encapsulation operators $\boxplus$ and $\boxminus$ and the ESIKF as defined in [32].

B. State representation

We use the IMU frame I as the body frame of reference and denote by G the global frame, which corresponds to the initial IMU frame. The state estimate at a time step k is represented by the following vector:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} {}^G\hat{\mathbf{R}}_{I,k}{}^T & {}^G\hat{\mathbf{p}}_{I,k}^T & {}^G\hat{\mathbf{v}}_{I,k}^T & \hat{\mathbf{b}}_{\boldsymbol{\omega},k}^T & \hat{\mathbf{b}}_{\mathbf{a},k}^T & {}^G\hat{\mathbf{g}}_{,k}^T \end{bmatrix}^T \in \mathrm{SO(3)} \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \quad (1)$$

where ${}^G\hat{\mathbf{R}}_{I,k}$ is the rotation, ${}^G\hat{\mathbf{p}}_{I,k}$ is the position, ${}^G\hat{\mathbf{v}}_{I,k}$ is the velocity of the body, and ${}^G\hat{\mathbf{g}}_{,k}$ is the gravity vector in the global frame. $\hat{\mathbf{b}}_{\boldsymbol{\omega},k}$ and $\hat{\mathbf{b}}_{\mathbf{a},k}$ are the gyroscope and accelerometer biases, expressed in the IMU frame. The error-state is written:

$$\begin{aligned}\delta\hat{\mathbf{x}}_k &= \mathbf{x}_k \boxminus \hat{\mathbf{x}}_k \\ &= \begin{bmatrix} {}^G\delta\hat{\mathbf{r}}_{I,k}^T & {}^G\delta\hat{\mathbf{p}}_{I,k}^T & {}^G\delta\hat{\mathbf{v}}_{I,k}^T & \delta\hat{\mathbf{b}}_{\boldsymbol{\omega},k}^T & \delta\hat{\mathbf{b}}_{\mathbf{a},k}^T & {}^G\delta\hat{\mathbf{g}}_k^T \end{bmatrix}^T \\ &\in \mathbb{R}^{18}\end{aligned} \quad (2)$$

where the $\boxminus$ operator implies that ${}^G\delta\hat{\mathbf{r}}_{I,k} = \mathrm{Log}(({}^G\hat{\mathbf{R}}_{I,k})^T {}^G\mathbf{R}_{I,k})$, ${}^G\delta\hat{\mathbf{p}}_{I,k} = {}^G\mathbf{p}_{I,k} - {}^G\hat{\mathbf{p}}_{I,k}$ and all other error-state quantities defined are defined similarly. The initial error-state covariance $\mathbf{P}_0$ is initialized as an identity matrix multiplied by a small number: $\mathbf{P}_0 = \varepsilon\mathbf{I}_{18\times18}$.

C. Propagation with IMU

Similarly to [21] and [23], we model the IMU input $\mathbf{u}_u$ with its noise and bias:

$$\mathbf{u} = \begin{bmatrix}\boldsymbol{\omega}^T & \mathbf{a}^T\end{bmatrix}^T \in \mathbb{R}^6, \quad (3)$$

$$\mathbf{w} = \begin{bmatrix}\mathbf{n}_{\boldsymbol{\omega}}^T & \mathbf{n}_{\mathbf{a}}^T & \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}}^T & \mathbf{n}_{\mathbf{b}_{\mathbf{a}}}^T\end{bmatrix}^T \in \mathbb{R}^{12} \quad (4)$$

where $\mathbf{n}_{\boldsymbol{\omega}}$ and $\mathbf{n}_{\mathbf{a}}$ are the Gaussian noises of the measurements from the gyroscope and accelerometer, and their respective biases $\mathbf{b}_{\boldsymbol{\omega}}, \mathbf{b}_{\mathbf{a}}$ are modeled as random walk processes with Gaussian noises $\mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}}$ and $\mathbf{n}_{\mathbf{b}_{\mathbf{a}}}$. Using Newton's dot notation for time derivatives, the kinematic model of the system in continuous time takes the form:

$${}^G\dot{\mathbf{R}}_I = {}^G\mathbf{R}_I[\boldsymbol{\omega} - \mathbf{b}_{\boldsymbol{\omega}} - \mathbf{n}_{\boldsymbol{\omega}}]_\times \quad (5)$$

$${}^G\dot{\mathbf{p}}_I = {}^G\mathbf{v}_I \quad (6)$$

$${}^G\dot{\mathbf{v}}_I = {}^G\mathbf{R}_I(\mathbf{a} - \mathbf{b}_{\mathbf{a}} - \mathbf{n}_{\mathbf{a}}) + {}^G\mathbf{g} \quad (7)$$

$$\dot{\mathbf{b}}_{\boldsymbol{\omega}} = \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}}}, \quad \dot{\mathbf{b}}_{\mathbf{a}} = \mathbf{n}_{\mathbf{b}_{\mathbf{a}}}, \quad {}^G\dot{\mathbf{g}} = \mathbf{0} \quad (8)$$

In discrete time, the state transition function between time steps k and $k+1$ is written: $\mathbf{x}_{k+1} = \mathbf{x}_k \boxplus (\Delta t\, g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k))$, where Δt is the time between consecutive IMU readings and the function g is defined as:

$$g(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) = \begin{bmatrix} \boldsymbol{\omega}_k - \mathbf{b}_{\boldsymbol{\omega},k} - \mathbf{n}_{\boldsymbol{\omega},k} \\ {}^G\mathbf{v}_{I,k} \\ {}^G\mathbf{R}_{I,k}(\mathbf{a}_k - \mathbf{b}_{\mathbf{a},k} - \mathbf{n}_{\mathbf{a},k}) + {}^G\mathbf{g}_k \\ \mathbf{n}_{\mathbf{b}_{\boldsymbol{\omega}},k} \\ \mathbf{n}_{\mathbf{b}_{\mathbf{a}},k} \\ \mathbf{0}_{3\times1} \end{bmatrix} \quad (9)$$

We refer the reader to [32] for the expressions of the error-state $\delta\hat{\mathbf{x}}_{k+1|k}$ and its covariance.

D. Correction with visual-inertial odometry

Our VIO sub-system is based on VINS-Mono [9], which we choose for its overall robustness to challenging scenarios, as we verify further in our experiments.

1) Observation model: Each time VINS-Mono returns a new pose of the IMU in the global frame ${}^G\mathbf{T}_{\mathrm{VIO},I,k+1} \in \mathrm{SE(3)}$, we compute the incremental pose change: ${}^{\mathrm{VIO},I,k}\mathbf{T}_{\mathrm{VIO},I,k+1} = {}^G\mathbf{T}_{\mathrm{VIO},I,k}^{-1}\, {}^G\mathbf{T}_{\mathrm{VIO},I,k+1}$. We then compute a "synthetic odometry observation" for step $k+1$ by combining this relative pose with the pose of the whole system output by the ESIKF at the k-th time step ${}^G\mathbf{T}_{I,k}$: ${}^G\mathbf{T}_{\mathrm{obs},I,k+1} = {}^G\mathbf{T}_{I,k}\, {}^{\mathrm{VIO},I,k}\mathbf{T}_{\mathrm{VIO},I,k+1}$. To process the SE(3) pose observation in the ESIKF update, we separate its SO(3) and $\mathbb{R}^3$ components:

$${}^G\mathbf{T}_{\mathrm{obs},I,k+1} = \begin{bmatrix} {}^G\mathbf{R}_{\mathrm{obs},I,k+1} & {}^G\mathbf{p}_{\mathrm{obs},I,k+1} \\ \mathbf{0} & 1 \end{bmatrix} \quad (10)$$

We model the observation noise as additive zero-mean Gaussian noise on both components with $\mathbf{n}_{\mathrm{VIO},k+1} = [\mathbf{n}_{r,k+1}^T \quad \mathbf{n}_{p,k+1}^T]^T \in \mathbb{R}^6$, with $\mathbf{n}_{\mathrm{VIO}k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{N}), \mathbf{N} \in \mathbb{R}^{6\times6}$. From here, we use the superscript j to indicate the current update iteration and drop the $k+1$ subscript to improve readability. The observation $\mathbf{Z}_{\mathrm{VIO}}^j \in \mathrm{SE(3)}$ and the observation function h_{VIO} are defined as:

$$\mathbf{Z}_{\mathrm{VIO}}^j = h_{\mathrm{VIO}}(\mathbf{x}, \mathbf{n}_{\mathrm{VIO}}) \quad (11)$$

$$= \begin{bmatrix} {}^G\mathbf{R}_I \boxplus \mathbf{n}_r & {}^G\mathbf{p}_I \boxplus \mathbf{n}_p \\ \mathbf{0} & 1 \end{bmatrix} \quad (12)$$

$$= \begin{bmatrix} {}^G\hat{\mathbf{R}}_I^j \boxplus {}^G\delta\hat{\mathbf{r}}_I^j \boxplus \mathbf{n}_r & \mathbf{0} \\ {}^G\hat{\mathbf{p}}_I^j \boxplus {}^G\delta\hat{\mathbf{p}}_I^j \boxplus \mathbf{n}_p & 1 \end{bmatrix}^T \quad (13)$$

$$= \begin{bmatrix} {}^G\hat{\mathbf{R}}_I^j \mathrm{Exp}({}^G\delta\hat{\mathbf{r}}_I^j)\mathrm{Exp}(\mathbf{n}_r) & \mathbf{0} \\ {}^G\hat{\mathbf{p}}_I^j + {}^G\delta\hat{\mathbf{p}}_I^j + \mathbf{n}_p & 1 \end{bmatrix}^T \quad (14)$$

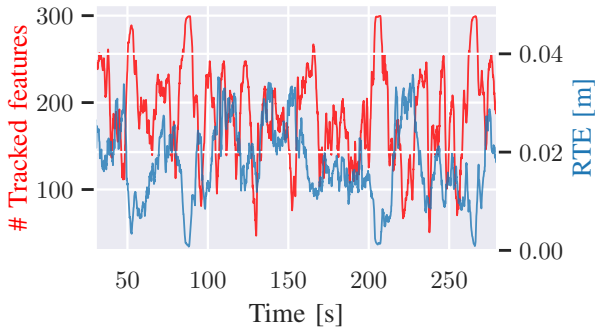

Fig. 5. Number of visual features tracked related to the relative translation error, shown in the active part of the Deg-Vis-1 sequence. The Pearson correlation coefficient is -0.31, indicating a moderate negative correlation.

The residual is defined as the $\boxminus$-difference between the actual measurement $\mathbf{Z}^j_{\text{VIO}}$ and the value of the measurement function with the noise set to zero:

$$\mathbf{r}^j_{\text{VIO}} = \mathbf{Z}^j_{\text{VIO}} \boxminus h(\hat{\mathbf{x}}^j, \mathbf{0}) \tag{15}$$

$$= h(\mathbf{x}, \mathbf{n}_{\text{VIO}}) \boxminus h(\hat{\mathbf{x}}^j, \mathbf{0}) \tag{16}$$

$$\approx \begin{bmatrix} {}^G\delta\hat{\mathbf{r}}^j_I + \mathbf{n}_r \\ ({}^G\hat{\mathbf{R}}^j_I)^T({}^G\delta\hat{\mathbf{p}}^j_I + \mathbf{n}_p) \end{bmatrix} \in \mathbb{R}^6 \tag{17}$$

From there, we compute a first order approximation to fuse the distribution imposed by the residual with the distribution of the propagated state:

$$\mathbf{r}^j_{\text{VIO}} \approx \mathbf{H}^j_{\text{VIO},\delta\mathbf{x}}\delta\hat{\mathbf{x}}^j + \mathbf{H}^j_{\text{VIO},\mathbf{n}}\mathbf{n}_{\text{VIO}} \tag{18}$$

where Jacobians can be derived by simple inspection of (17).

2) Adaptative covariance estimation: By default, VINS-Mono does not output any covariance with its pose estimate, as in practice there is no way to compute it in a manner both fast and reliable[5]. To this end, we propose an adaptative covariance heuristic based on the number of tracked features. As illustrated in Figure 5, this number can be used as an online indicator for the current odometry accuracy. Especially when the maximum number of features is tracked in a stable manner (flat-top peaks in the figure), the relative translation error (RTE) is at a minimum. We start with a default covariance matrix in the form of $\mathbf{V}^* = \varepsilon_{\text{VIO}}\mathbf{I}_{6\times 6}$ where ε_{VIO} is a number entered as a parameter. We find that to ensure the VIO update has an influence during the ESIKF update, ε_{VIO} should be set to a very small value (e.g. 10^{-6}), in order to balance the weight of the VIO residual with that of the all the point-to-plane residuals. Then, we use the number of tracked features in the current frame F_{k+1} and the target number of features to track F^* (a parameter of VINS-Mono, here set to 300). At each frame, the proposed covariance $\mathbf{V}_{k+1}$ is computed with:

$$\mathbf{V}_{k+1} = \left(1 + \frac{F^* - F_{k+1}}{\gamma}\right)\mathbf{V}^* \tag{19}$$

where $\gamma \in \mathbb{R}_{>0}$ is a parameter determining the level of covariance increase due to a sub-optimal number of tracked features. In Experiment I, we show the influence of tuning the parameter γ on the performance of the whole system.

[5]VINS-Mono uses a nonlinear least squares problem formulation, solved with Ceres [33]. See http://ceres-solver.org/nnls_covariance.html.

E. Correction with point cloud odometry and mapping

1) Observation model: Our Depth Odometry and Mapping (DOM) sub-system follows the implementation of [3]. Each voxel contains the parameters of a probabilistic plane element (normal and its covariance, center and its covariance). Each point of each new scan is matched against the current map. For the i-th point-to-plane match, the observation model function $h_{\text{DOM,i}}$ yields the point-to-plane distance $z^j_{\text{DOM}i} \in \mathbb{R}$ and is defined as:

$$z^j_{\text{DOM}i} = h_{\text{DOM,i}}(\mathbf{x}, \mathbf{n}_{\text{DOM},i}) = \mathbf{n}_i^T({}^G\mathbf{T}^j_I\mathbf{s}_i - \mathbf{t}_i) \tag{20}$$

$$= (\hat{\mathbf{n}}_i \boxplus \delta\hat{\mathbf{n}}_i)^T((^G\hat{\mathbf{T}}^j_I \boxplus {}^G\delta\hat{\mathbf{T}}^j_I)(\hat{\mathbf{s}}_i + \delta\hat{\mathbf{s}}_i) - \hat{\mathbf{t}}_i - \delta\hat{\mathbf{t}}_i) \tag{21}$$

$$= (\hat{\mathbf{n}}_i \boxplus \delta\hat{\mathbf{n}}_i)^T\Big((^G\hat{\mathbf{R}}^j_I \boxplus {}^G\delta\hat{\mathbf{r}}^j_I)(\hat{\mathbf{s}}_i + \delta\hat{\mathbf{s}}_i) + ({}^G\hat{\mathbf{p}}^j_I + {}^G\delta\hat{\mathbf{p}}^j_I) - \hat{\mathbf{t}}_i - \delta\hat{\mathbf{t}}_i\Big) \tag{22}$$

where $\mathbf{n}_{\text{DOM},i} = [\delta\hat{\mathbf{n}}_i^T\ \delta\hat{\mathbf{t}}_i^T\ \delta\hat{\mathbf{s}}_i^T]^T \in \mathbb{R}^9$ encodes the noise of the observation, as $\hat{\mathbf{n}}_i$ and $\delta\hat{\mathbf{n}}_i$ are the plane normal and its associated noise, $\hat{\mathbf{t}}_i$ and $\delta\hat{\mathbf{t}}_i$ are the plane center and its associated noise, and finally $\hat{\mathbf{s}}_i$ and $\delta\hat{\mathbf{s}}_i$ are the point and its associated noise. The expression of the residual becomes as follows:

$$r^j_{\text{DOM},i} = z^j_{\text{DOM},i} \boxminus h_{\text{DOM,i}}(\hat{\mathbf{x}}^j, \mathbf{0}) \tag{23}$$

$$= \mathbf{n}_i^T({}^G\mathbf{T}^j_I\mathbf{s}_i - \mathbf{t}_i) - \hat{\mathbf{n}}_i^T({}^G\hat{\mathbf{T}}^j_I\hat{\mathbf{s}}_i - \hat{\mathbf{t}}_i) \tag{24}$$

$$= (\hat{\mathbf{n}}_i \boxplus \delta\hat{\mathbf{n}}_i)^T\Big((^G\hat{\mathbf{R}}^j_I \boxplus {}^G\delta\hat{\mathbf{r}}^j_I)(\hat{\mathbf{s}}_i \boxplus \delta\hat{\mathbf{s}}_i) + ({}^G\hat{\mathbf{p}}^j_I \boxplus {}^G\delta\hat{\mathbf{p}}^j_I) - (\hat{\mathbf{t}}_i + \delta\hat{\mathbf{t}}_i)\Big) - \hat{\mathbf{n}}_i^T({}^G\hat{\mathbf{T}}^j_I\hat{\mathbf{s}}_i - \hat{\mathbf{t}}_i) \tag{25}$$

$$\approx \mathbf{H}^j_{\text{DOM,i},\delta\mathbf{x}}\delta\hat{\mathbf{x}}^j + \mathbf{H}^j_{\text{DOM,i},\mathbf{n}}\mathbf{n}_{\text{DOM},i} \in \mathbb{R} \tag{26}$$

2) Degeneracy handling with soft solution remapping: It is known that the performance of depth cameras can be strongly degraded in various environments [34]. Often, this degradation will lead to fewer points returned in each depth map, and in turn, fewer geometrical features. This may increase degeneracy and drift. To mitigate these issues, we implement the solution remapping method described in [6], which proposes to analyse the eigenvalues of the product $\mathbf{H}^T\mathbf{H} \in \mathbb{R}^{n\times n}$, where $\mathbf{H} = [(\mathbf{H}^j_{\text{DOM,1},\delta\mathbf{x}})^T, (\mathbf{H}^j_{\text{DOM,2},\delta\mathbf{x}})^T, \ldots, (\mathbf{H}^j_{\text{DOM,m},\delta\mathbf{x}})^T]^T$, m is the number of residuals and n is the dimensionality of the state. If the i-th eigenvalue is below a certain threshold ζ set as user-input parameter, the update in the i-th direction of the eigenspace is discarded. Using the notation $\mathbf{e}_1, ..., \mathbf{e}_n$ for the eigenvectors corresponding to the eigenvalues $\lambda_1, ..., \lambda_n$ of $\mathbf{H}^T\mathbf{H}$, the remapped solution $\delta\hat{\mathbf{x}}^j_{SR}$ is determined with:

$$\mathbf{E}_f = [\mathbf{e}_1, ..., \mathbf{e}_m, \mathbf{e}_{m+1}, ..., \mathbf{e}_n]^T \tag{27}$$

$$\mathbf{E}_u = [\mathbf{0}, ..., \mathbf{0}, \mathbf{e}_{m+1}, ..., \mathbf{e}_n]^T \tag{28}$$

$$\delta\hat{\mathbf{x}}^j_{SR} = \mathbf{E}_f^{-1}\mathbf{E}_u\delta\hat{\mathbf{x}}^j \tag{29}$$

where m is the number of eigenvalues smaller than ζ (with $0 \leq m \leq n$), $\mathbf{E}_f$ is a matrix whose rows are the eigenvectors of $\mathbf{H}^T\mathbf{H}$, and $\mathbf{E}_u$ is the same matrix but the eigenvectors corresponding to eigenvalues smaller than ζ are set to zero. Tuning the parameter ζ is commonly done by hand. Now, through experimentation we find that this binary thresholding is a rather strict measure which can make the parameter tuning

process difficult: a slightly too high ζ can quickly lead to significantly worse solutions. To cope with this, we propose a soft version of this update by modifying $\mathbf{E}_u$:

$$\mathbf{E}_u = [\frac{\lambda_1}{\zeta}\mathbf{e}_1, ..., \frac{\lambda_m}{\zeta}\mathbf{e}_m, \mathbf{e}_{m+1}, ..., \mathbf{e}_n]^T \tag{30}$$

We show in Experiment II that this soft solution remapping has a positive effect on the accuracy of the system. Finally, we propose an automated initialization procedure for tuning ζ: when the system starts, we collect all eigenvalues for a duration of t_{init}, set to 10s in our experiments. Making the assumption that there is no degeneracy in the beginning of the sequences until t_{init}, the eigenvalues collected must be higher than our desired threshold. After t_{init}, we take the minimum eigenvalue and set ζ to its value multiplied by 10^{-2} for a safety margin, avoiding to discard useful updates.

V. Experiments

We evaluate our system using the RMS absolute translation error (ATE) as main criterion. All of our experiments are executed on an ASUS NUC 13 Pro NUC13ANHi7 with an Intel Core i7-1360P CPU.

A. Experiment I: VIO adaptative covariance estimation

TABLE III
Accuracy (ATE) of our method with different γ values

Sequence \ γ-value	1	10	100	1000	∞
Deg-Vis-1	1.108	0.716	**0.579**	0.660	0.620
Deg-Vis-2	0.550	0.546	**0.411**	0.447	0.448

We start by experimenting with the value of γ as described in Section IV-D. We run our complete system on the two visually degenerate sequences of the Bunker DVI Dataset with γ-values ranging from 1 (large penalty for low numbers of tracked features) to ∞ (no penalty at all). We see that among the γ values sampled, we achieve best results with $\gamma = 100$, which means that in the extreme case of having no feature tracked at all, the proposed covariance is $\mathbf{V}_{k+1} = (1 + \frac{300-0}{100})\mathbf{V}* = 4\mathbf{V}*$.

B. Experiment II: Degeneracy handling with solution remapping

To evaluate our soft solution remapping (SSR) component, we run the method on our depth-inertial pipeline (without VIO fusion), varying the ζ-value and comparing against the hard solution remapping (HSR). Results of this experiments are displayed in Figure 6, and show that the proposed SSR effectively enlarges the basin of performance-improving values for the parameter ζ. Next, we test our auto-threshold tuning procedure. In Figure 7, we report the tests of our system on all six sequences of the Bunker DVI Dataset in three configurations: no SR, HSR and SSR (both with auto-threshold). Without surprise, we see that neither HSR or SSR affects the non-degenerate sequences (Reg-1 until Deg-Vis-2). Then, we notice that the auto-SSR positively effects the accuracy of the system in the geometrically degenerate sequences Deg-Geo-1 and Deg-Geo-2. What is more, in the Deg-Geo-2 sequence, the auto-HSR fails to decrease the error (likely due to a too high ζ for this sequence), but auto-SSR manages to alleviate the issue and improves the performance over the baseline.

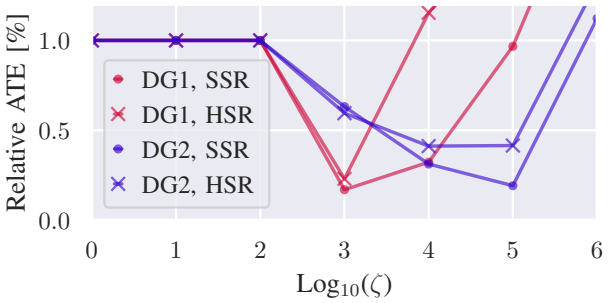

Fig. 6. Relative ATE (ATE with the given ζ-value / ATE without SR) for different ζ-values ranging from 1 to 10^6. DG* refers to the Deg-Geo-* sequences, and SSR / HSR to soft and hard SR.

Fig. 7. ATE of our system with different solution remapping (SR) methods.

C. Experiment III: Comparison with the state-of-the-art and ablation study

This experiment aims to evaluate the performance of DVI odometry and mapping algorithms in the three categories (RGB-D, stereo and point cloud-based) and depth sensing modalities (ToF, active stereo (AS) and passive stereo (PS)). We use the three DVI sensors available in the Bunker DVI Dataset: Microsoft Kinect for Azure (K4A), Intel Realsense D455F (D455F) and Sevensense Core Research (7S)[6]. We evaluate both industry standards (e.g. VINS-Mono [9], RTABMap [10]) and more recent state-of-the-art methods (such as VINS-RGBD-FAST [11] (referred to as VINS-RGBD-F), OpenVINS [12]). These methods are chosen for their state-of-the-art performance (robustness, precision and speed) on portable computing systems (without GPU). Note that we exclude methods based on deep learning and neural radiance fields (NeRFs) as they require a GPU and are notoriously slower and less robust, especially in uncontrolled and perturbed environments as we consider here. Whenever possible, we use the parameters provided by the authors of the method for the sensor. Otherwise, we tune the parameters on the calibration and warm-up sequences. Table IV summarizes our results.

[6]The implementation of passive depth stereo is based on https://github.com/sevensense-robotics/alphasense_stereo_demo

TABLE IV
ATE [M] OF THE STATE-OF-THE-ART METHODS AND ABLATION STUDY OF OUR PROPOSED SYSTEM ON THE BUNKER DVI DATASET

Sequence	**Regular-1**			**Regular-2**			**Deg-Vis-1**			**Deg-Vis-2**			**Deg-Geo-1**			**Deg-Geo-2**		
Device	K4A	D455F	7S	K4A	D455F	7S	K4A	D455F	7S	K4A	D455F	7S	K4A	D455F	7S	K4A	D455F	7S
Depth tech.	ToF	AS	PS	ToF	AS	PS	ToF	AS	PS	ToF	AS	PS	ToF	AS	PS	ToF	AS	PS
IMU-RGB																		
VINS-Mono	4.59	2.79	0.97	9.90	7.07	2.06	6.43	6.33	2.16	5.15	5.60	25.52	4.73	3.57	**1.26**	1.29	1.68	**0.34**
OpenVINS	8.21	39.71	4.38	F1	38.35	6.48	F1	18.52	4.15	F1	F2	F2	F1	18.86	69.09	F1	6.66	1.37
IMU-RGB-D																		
VINS-RGBD-F	6.12	3.54	1.39	2.49	8.39	3.78	20.42	6.37	10.15	6.94	4.48	31.22	5.47	7.61	<u>1.89</u>	3.80	1.49	<u>0.67</u>
RTABMap	8.19	8.39	0.37	45.40	22.01	F3	16.20	31.70	F3	31.46	11.04	F3	17.77	42.71	30.23	9.39	10.58	7.07
IMU-Stereo																		
OpenVINS	-	-	4.65	-	-	4.09	-	-	3.26	-	-	F2	-	-	3.05	-	-	1.68
VINS-Fusion	-	-	3.52	-	-	5.15	-	-	F4	-	-	4.86	-	-	4.51	-	-	2.89
IMU-Cloud																		
RTABMap	0.36	F5	F5	0.81	17.37	F5	1.96	F5	F5	<u>0.48</u>	F5	F5	9.06	F5	F5	4.14	F5	F5
VoxelMap*	0.26	F5	F5	<u>0.26</u>	F5	F5	1.32	F5	F5	0.95	F5	F5	8.76	F5	F5	10.32	F5	F5
Ours (D)*	<u>0.25</u>	F5	F5	0.27	F5	F5	1.34	F5	F5	0.95	F5	F5	5.68	F5	F5	8.19	F5	F5
Ours (DI)	**0.22**	F5	F5	**0.22**	F5	F5	<u>0.75</u>	F5	F5	1.70	F5	F5	10.64	F5	F5	6.32	F5	F5
IMU-RGB-Cloud																		
Ours (DVI)	0.28	F5	F5	0.28	F5	F5	**0.63**	F5	F5	**0.46**	F5	F5	4.00	F5	F5	1.44	F5	F5

The symbol "-" indicates that the method is not evaluated for this sensor, i.e. in the case of a stereo-based method with time-of-flight sensors (no stereo pair) or active stereo depth sensors (left and right images show the IR pulses, making conventional stereo matching impossible). "*" indicates that the IMU is not used. Bold and underline fonts indicate the per-sequence best and second-best result. The F1 to F5 indicate the type of failure, explained in greater detail in the text below.

General remarks. First, we notice that RGB, RGB-D and stereo methods are generally more robust (fewer total failures) than cloud-based methods. However, in the cases where the cloud-based methods do work, they are significantly more precise. As a side note, this further justifies our choice of fusing VINS-Mono (the most robust visual-inertial system, as it does not crash in any sequence) with VoxelMap (most precise depth-based system, until geometrical degeneracies are found). We also notice that, in this challenging dataset, the simple but well-engineered VINS-Mono often performs better than the more complex RGB-D or stereo methods. Finally, in cases where visual degeneracy is not too important, we see that the methods evaluated using data from the 7S device often yield excellent results, which can be attributed to the specific VIO-related characteristics of this sensor (such as precise time synchronization between IMU and cameras, global shutter sensors and wide angle lenses).

Study of failure cases. During each evaluation, we take note of the failure events, classify them and summarize them using the F1, . . ., F5 in the table:

- F1: Tracking lost during fast motion (also, the algorithm does not model rolling shutter RGB cameras as in K4A)
- F2: Tracking was lost after an abrupt light change.
- F3: The depth map from passive depth was very noisy. (Degraded lighting conditions and flat walls negatively impact the passive depth accuracy.)
- F4: Divergence happened when the person was walking close to the front of the sensors, creating too many unstable features.
- F5: The point clouds are too noisy for registration.

Unreported results. In addition to the successful executions reported in the table, we also experimented with ORB-SLAM3 [8], Fast-LIO [21], SLICT [18], VoxelMap++ [35], but we could not make these methods work for any of the evaluation sequences, either due to tracking loss, divergence or crash.

Performance of the DVI Mapping System. Although shown to work only with the ToF DVI sensor, we see that our proposed system achieves the most robust results on the dataset. It successfully estimates the full trajectory in all sequences and yields either the best accuracy or a competitive one.

TABLE V
TIME TO PROCESS ONE FRAME, IN MILLISECONDS.

DOM	SR	IMU	VIO	Frame processing time [ms]
×				64.2
			×	35.8
×	×			66.3
×		×		66.8
×		×	×	96.1
×	×	×		69.5
×	×	×	×	98.7

On Table V, we report the average time to process one frame in each DVI Mapping System configuration. Clearly, the two main sub-systems (DOM and VIO) take up the most processing time. The IMU propagation and solution remapping both add only a few milliseconds of processing time. The complete system takes just below 100ms to process a frame, meaning that it would run in real-time on a 10 FPS dataset.

VI. CONCLUSION

In this paper we have developed an error-state iterative Kalman filter (ESIKF) sensor fusion framework for depth-visual-inertial (DVI) 3D mapping. The system specifically targets small form-factor time-of-flight depth-visual-inertial

sensors, which are not supported by major state-of-the-art LiDAR mapping algorithms. The framework has been implemented with open-source technologies, developed in C++ with the ROS framework. We have conducted experiments to assess the performance and effectiveness of our sensor fusion scheme on a new challenging dataset. Our results suggest that our system may provide comparable or superior performance to existing methods designed for similar sensors. Following open science principles, we release our code and dataset publicly.

REFERENCES

[1] Livox, "Livox Avia," https://www.livoxtech.com/avia, 2024, [Online; accessed 17-March-2024].

[2] H. Wang, C. Wang, and L. Xie, "Lightweight 3-d localization and mapping for solid-state lidar," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1801–1807, 2021.

[3] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robotics and Automation Letters*, vol. 7, pp. 8518–8525, 2022.

[4] Microsoft, "Azure Kinect DK," https://azure.microsoft.com/en-us/products/kinect-dk, 2024, [Online; accessed 17-March-2024].

[5] Intel, "Realsense L515," https://www.intelrealsense.com/lidar-camera-l515/, 2024, [Online; accessed 17-March-2024].

[6] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 809–816.

[7] Hamesse, Charles and Luong, Hiep and Haelterman, Rob, "The state of depth sensors and depth estimation algorithms for dense 3d reconstruction," 2022.

[8] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

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

[10] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[11] J. Liu, X. Li, Y. Liu, and H. Chen, "Rgb-d inertial odometry for a resource-restricted robot in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9573–9580, 2022.

[12] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. P. Huang, "Openvins: A research platform for visual-inertial estimation," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4666–4672, 2020.

[13] V. C. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, pp. 422–429, 2019.

[14] J. Mo and J. Sattar, "Continuous-time spline visual-inertial odometry," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9492–9498, 2021.

[15] M. Vlaminck, H. Luong, W. Goeman, P. Veelaert, and W. Philips, "Towards online mobile mapping using inhomogeneous lidar data," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2016, pp. 845–850.

[16] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

[17] T. Shan, B. Englot, C. Ratti, and D. Rus, "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5692–5698, 2021.

[18] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "Slict: Multi-input multi-scale surfel-based lidar-inertial continuous-time odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2102–2109, 2023.

[19] T.-M. Nguyen, X. Xu, T. Jin, Y. Yang, J. Li, S. Yuan, and L. Xie, "Eigen is all you need: Efficient lidar-inertial continuous-time odometry with internal association," *IEEE Robotics and Automation Letters*, vol. 9, pp. 5330–5337, 2024.

[20] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4861–4868, 2022.

[21] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[22] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[23] J. Lin, C. Zheng, W. Xu, and F. Zhang, "R ^{2} live: A robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7469–7476, 2021.

[24] J. Lin and F. Zhang, "R3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 672–10 678.

[25] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[26] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[27] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[28] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, Y. Guo, Z. Wang, Y. Zhang, B. Qin, W. Yang, F. Wang, R. H. M. Chan, and Q. She, "Are we ready for service robots? the OpenLORIS-Scene datasets for lifelong SLAM," in *2020 International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3139–3145.

[29] Femto, "Bolt," https://www.orbbec.com/products/tof-camera/femto-bolt/, 2024, [Online; accessed 17-September-2024].

[30] T.-M. Nguyen, S. Yuan, T. H. Nguyen, P. Yin, H. Cao, L. Xie, M. Wozniak, P. Jensfelt, M. Thiel, J. Ziegenbein, and N. Blunder, "Mcd: Diverse large-scale multi-campus dataset for robot perception," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6 2024. [Online]. Available: https://mcdviral.github.io/

[31] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. F. Fallon, "The newer college dataset: Handheld lidar, inertial and vision with ground truth," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4353–4360, 2020.

[32] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," 2021.

[33] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[34] E. Curto and H. Araujo, "An experimental assessment of depth estimation in transparent and translucent scenes for intel realsense d415, sr305 and l515," *Sensors*, vol. 22, no. 19, 2022.

[35] C. Wu, Y. You, Y. Yuan, X. jiang Kong, Y. Zhang, Q. Li, and K. Zhao, "Voxelmap++: Mergeable voxel mapping method for online lidar(-inertial) odometry," *IEEE Robotics and Automation Letters*, vol. 9, pp. 427–434, 2023.