

Depth-Visual-Inertial (DVI) Mapping for Large-Scale Indoor 3D Reconstruction

Charles Hamesse^{*†}, Michiel Vlamincx[†], Hiep Luong[†], Rob Haelterman^{*}

^{*} Royal Military Academy, Department of Mathematics, Brussels, Belgium

[†] Ghent University, imec - IPI - URC, Ghent, Belgium

Abstract—We propose the Depth-Visual-Inertial (DVI) mapper: 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 RGB-D sensors. We propose a new challenging dataset for evaluation and 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—3D mapping, depth camera, time-of-flight sensor

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 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) mapper, 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 with a novel adaptive covariance estimation method, and finally implement a solution remapping algorithm to discard updates in degenerate directions of the state space, as proposed initially in [6]. For evaluation, we collect a new dataset with Microsoft Azure Kinect [4] and Intel Realsense L515 [5]. Our results show that our method is a viable option for real-life mapping scenarios in complex indoor environments. We open source our code and datasets¹.

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 maps (as shown in Table I). 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. In our experiments, we use the Intel Realsense L515 and the Microsoft Azure Kinect DK [4].

b) *LiDAR(-inertial) odometry and mapping*: Recent methods for dense 3D mapping with LiDARs achieve very robust and accurate results [8]–[14]. Most state-of-the-art mapping systems rely on some variant of the Kalman filter such as the ESIKF [15], and a specific 3D map representation tailored to the specific use case [14], [16], [17]. These methods accomplish great performance, even with solid-state LiDARs. However, we fail to see developments of similar maturity in the case of depth cameras. In fact, recent state-of-the-art LiDAR-based 3D mapping methods such as FAST-LIO [16], [17], RxLive [18], [19] or LVI-SAM [11] have not been successfully applied to depth cameras. This can be attributed to the noisier and scarcer nature of the depth maps

This work was funded in part by Belgium's Royal Higher Institute for Defence (RHID) under Grant DAP18/04.

¹The project's webpage can be found at <https://charleshamesse.github.io/dvi-mapper-web/>

TABLE I
MAIN SPECIFICATIONS OF 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
Azure Kinect DK	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

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, a DVI sensor with ToF camera. SSL-SLAM implements a scan-to-map registration method operating on edge and planar features to increase the tracking performance, optimized with a sliding window. VoxelMap implements scan-to-map registration with an ESIKF and a fully probabilistic voxel octree, where each voxel contains the parameters of a plane element, including covariance updates arising from both point measurement noise and pose estimation error, which are then used in the state update. Note that RGB-D frameworks like ORB-SLAM3 [20] rely on stereo or depth only to initialize visual features, without using dense depth information. Moreover, both sensors mentioned previously (Intel Realsense L515 and the Microsoft Azure Kinect) have a global shutter depth camera but a rolling shutter RGB camera, which makes feature matching between RGB and depth frames more difficult.

c) *Visual-inertial odometry (VIO) and mapping*: Although major improvements in learning-based methods for visual(-inertial) odometry have been proposed [21]–[23], the most precise and robust methods remain the classical optimization frameworks such as VINS-Mono [24], ORB-SLAM [20] and more recently OpenVINS [25] and Basalt [26]. This is especially true in complex cases with poorly textured environments or fast rotations [27]. All of these systems are tightly-coupled systems using either sliding window optimization or iterative Kalman filtering. In [26], [28], B-spline continuous trajectory representations are used in order to perform more precise sensor fusion, at the cost of some computational expense.

Conceptually, our work shares similarities with R2Live [18] or R3Live [19]. They are visual extensions of a LiDAR-inertial method, our work is a visual-inertial extension of a ToF 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.

III. METHOD

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 $\|\mathbf{x}\|_{\Sigma} = \sqrt{\mathbf{x}^T \Sigma^{-1} \mathbf{x}}$. The skew-symmetric matrix of a vector of a vector $\mathbf{v} \in \mathbb{R}^3$ is noted $[\mathbf{v}]_{\times}$. We make use of the Lie groups $\text{SO}(3)$ and $\text{SE}(3)$, the manifold encapsulation operators $\boxplus$ and $\boxminus$ and the ESIKF as defined in [15].

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 of the system at time k is represented by the following vector:

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

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 ${}^G\hat{\mathbf{g}}_{I,k}$ is the gravity vector in the global frame. $\hat{\mathbf{b}}_{\omega,k}$ and $\hat{\mathbf{b}}_{\mathbf{a},k}$ are the gyroscope and accelerometer biases, expressed in the IMU frame.

C. Propagation with IMU

We implement the IMU initialization, pre-integration and state transition methods developed in [17], [18], [24], [29].

D. Correction with Visual-Inertial Odometry

Our VIO sub-system is based on VINS-Mono [24], which we choose for its robust state-of-the-art performance and its rolling shutter compensation component. Each time VINS-Mono returns a new pose ${}^G\mathbf{T}_{\text{VIO},I,k+1} \in \text{SE}(3)$, we compute the incremental pose change: ${}^{I,k}\mathbf{T}_{\text{VIO},I,k+1} = {}^G\mathbf{T}_{\text{VIO},I,k}^{-1} {}^G\mathbf{T}_{\text{VIO},I,k+1}$. We then compute a “synthetic odometry observation” for step $k+1$ by combining this relative pose with the pose of the whole system in its state at the k -th time step ${}^G\mathbf{T}_{I,k}$: ${}^G\mathbf{T}_{\text{obs},I,k+1} = {}^G\mathbf{T}_{I,k} {}^{I,k}\mathbf{T}_{\text{VIO},I,k+1}$. We now perform an iterated state correction to fuse the information of this observation with the prior distribution obtained in the previous step. To process the $\text{SE}(3)$ pose observation, we separate its $\text{SO}(3)$ and $\mathbb{R}^3$ components:

$${}^G\mathbf{T}_{\text{obs},I,k+1} = \begin{bmatrix} {}^G\mathbf{R}_{\text{obs},I,k+1} & {}^G\mathbf{p}_{\text{obs},I,k+1} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2)$$

We model the observation noise as additive zero-mean Gaussian noise on both components with $\mathbf{v}_{\text{VIO},k+1} = [\mathbf{v}_{r,k+1}^T \ \mathbf{v}_{p,k+1}^T]^T \in \mathbb{R}^6$, with $\mathbf{v}_{\text{VIO},k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$, $\mathbf{V} \in \mathbb{R}^{6 \times 6}$. The observation $\mathbf{Z}_{\text{VIO},k+1}^j \in \text{SE}(3)$ and the observation function h_{VIO} are defined accordingly:

$$\mathbf{Z}_{\text{VIO},k+1}^j = h_{\text{VIO}}(\mathbf{x}_{k+1}, \mathbf{v}_{\text{VIO},k+1}) \quad (3)$$

$$= \begin{bmatrix} {}^G\mathbf{R}_{I,k+1} \boxplus \mathbf{v}_{r,k+1} & {}^G\mathbf{p}_{I,k+1} \boxplus \mathbf{v}_{p,k+1} \\ \mathbf{0} & 1 \end{bmatrix} \quad (4)$$

$$= \begin{bmatrix} {}^G\hat{\mathbf{R}}_I^j \boxplus {}^G\delta\hat{\mathbf{r}}_I^j \boxplus \mathbf{v}_{r,k+1} & \mathbf{0} \\ {}^G\hat{\mathbf{p}}_{I,k+1} \boxplus {}^G\delta\hat{\mathbf{p}}_I^j \boxplus \mathbf{v}_{p,k+1} & 1 \end{bmatrix}^T \quad (5)$$

$$= \begin{bmatrix} {}^G\hat{\mathbf{R}}_I^j \text{Exp}({}^G\delta\hat{\mathbf{r}}_{I,k+1}) \text{Exp}(\mathbf{v}_{r,k+1}) & \mathbf{0} \\ {}^G\hat{\mathbf{p}}_I^j + {}^G\delta\hat{\mathbf{p}}_{I,k+1} + \mathbf{v}_{p,k+1} & 1 \end{bmatrix}^T \quad (6)$$

The residual is defined as the $\boxminus$ -difference between the actual measurement $\mathbf{Z}_{\text{VIO},k+1}^j$ and the value of the measurement function with the noise set to zero:

$$\mathbf{r}_{\text{VIO},k+1}^j = \mathbf{Z}_{\text{VIO},k+1}^j \boxminus h(\hat{\mathbf{x}}^j, \mathbf{0}) \quad (7)$$

$$= h(\mathbf{x}_{k+1}, \mathbf{v}_{\text{VIO},k+1}) \boxminus h(\hat{\mathbf{x}}^j, \mathbf{0}) \quad (8)$$

$$\approx \begin{bmatrix} {}^G\delta\hat{\mathbf{r}}_I^j + \mathbf{v}_{r,k+1} \\ ({}^G\hat{\mathbf{R}}_I^j)^T ({}^G\delta\hat{\mathbf{p}}_I^j + \mathbf{v}_{p,k+1}) \end{bmatrix} \in \mathbb{R}^6 \quad (9)$$

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

$$\mathbf{r}_{\text{VIO},k+1}^j \approx \mathbf{H}_{\text{VIO},\delta\mathbf{x},k+1}^j \delta\hat{\mathbf{x}}^j + \mathbf{H}_{\text{VIO},\mathbf{v},k+1}^j \mathbf{v}_{\text{VIO},k+1} \quad (10)$$

where Jacobians can be derived by simple inspection of (9).

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 reliable². To this end, we propose an adaptative covariance heuristic based on the number of tracked features. The motivation is that more features tracked in the current frame will lead to a more accurate pose estimation, and vice versa. Starting with a default covariance matrix in the form of $\mathbf{V}^* = \varepsilon_{\text{VIO}} \mathbf{I}_{6 \times 6}$ where ε_{VIO} is a small number entered as a parameter, 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 by default in VINS-Mono). At each frame, $\mathbf{V}_{k+1}$ is multiplied at each frame with:

$$\mathbf{V}_{k+1} = \left(1 + \frac{F^* - F_{k+1}}{\gamma}\right) \mathbf{V}^* \quad (11)$$

where $\gamma \in \mathbb{R}_{>0}$ is a parameter. In our experiments, we use $\gamma = 10$. Also, note that in any case $F^* \geq F_{k+1}$.

E. Correction with Depth Odometry and Mapping

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

²VINS-Mono uses a nonlinear least squares problem formulation, solved with Ceres [30]. See http://ceres-solver.org/nnls_covariance.html.



(a) Lixov Avia (top) and Azure Kinect (bottom)



(b) Ouster OS1-128 (top) and L515 (right)

Fig. 1. Reference data collection systems.

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 $z_{\text{DOM},i,k+1}^j \in \mathbb{R}$ and is defined as:

$$z_{\text{DOM},i,k+1}^j = h_{\text{DOM},i}(\mathbf{x}_{k+1}, \mathbf{v}_{\text{DOM},k+1}) = \mathbf{n}_i^T ({}^G\mathbf{T}_I^j \mathbf{s}_i - \mathbf{t}_i) \quad (12)$$

$$= (\hat{\mathbf{n}}_i \boxplus \delta\hat{\mathbf{n}}_i)^T (({}^G\hat{\mathbf{T}}_I^j \boxplus {}^G\delta\hat{\mathbf{T}}_I^j)(\hat{\mathbf{s}}_i + \delta\hat{\mathbf{s}}_i) - \hat{\mathbf{t}}_i - \delta\hat{\mathbf{t}}_i) \quad (13)$$

$$= (\hat{\mathbf{n}}_i \boxplus \delta\hat{\mathbf{n}}_i)^T \left(({}^G\hat{\mathbf{R}}_I^j \boxplus {}^G\delta\hat{\mathbf{r}}_I^j)(\hat{\mathbf{s}}_i + \delta\hat{\mathbf{s}}_i) + ({}^G\hat{\mathbf{p}}_I^j + {}^G\delta\hat{\mathbf{p}}_I^j) - \hat{\mathbf{t}}_i - \delta\hat{\mathbf{t}}_i \right) \quad (14)$$

where $\mathbf{v}_{\text{DOM},k+1} = [\delta\hat{\mathbf{n}}_i^T \ \delta\hat{\mathbf{t}}_i^T \ \delta\hat{\mathbf{s}}_i^T]^T \in \mathbb{R}^9$ 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:

$$r_{\text{DOM},k+1}^j = z_{\text{DOM},k+1}^j \boxminus h(\hat{\mathbf{x}}^j, \mathbf{0}) \quad (15)$$

$$= \mathbf{n}_i^T ({}^G\mathbf{T}_I^j \mathbf{s}_i - \mathbf{t}_i) - \hat{\mathbf{n}}_i^T ({}^G\hat{\mathbf{T}}_I^j \hat{\mathbf{s}}_i - \hat{\mathbf{t}}_i) \quad (16)$$

$$= (\hat{\mathbf{n}}_i \boxplus \delta\hat{\mathbf{n}}_i)^T \left(({}^G\hat{\mathbf{R}}_I^j \boxplus {}^G\delta\hat{\mathbf{r}}_I^j)(\hat{\mathbf{s}}_i \boxplus \delta\hat{\mathbf{s}}_i) + ({}^G\hat{\mathbf{p}}_{I,i} \boxplus {}^G\delta\hat{\mathbf{p}}_{I,i}) - (\hat{\mathbf{t}}_i + \delta\hat{\mathbf{t}}_i) \right) - \hat{\mathbf{n}}_i^T ({}^G\hat{\mathbf{T}}_I^j \hat{\mathbf{s}}_i - \hat{\mathbf{t}}_i) \quad (17)$$

$$\approx \mathbf{H}_{\text{DOM},\delta\mathbf{x},k+1}^j \delta\hat{\mathbf{x}}^j + \mathbf{H}_{\text{DOM},\mathbf{v},k+1}^j \mathbf{v}_{\text{DOM},k+1} \in \mathbb{R} \quad (18)$$

It is known that the performance of depth cameras can be strongly degraded in various environments [31]. 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 $\mathbf{H}_{\text{DOM},\delta\mathbf{x},k+1}^T \mathbf{H}_{\text{DOM},\delta\mathbf{x},k+1}$. If the i -th eigenvalue is below a certain threshold set as user-input parameter ζ , then we discard the error-state update in the i -th direction. In our experiments, we use $\zeta = 20000$.

IV. EVALUATION

Given the difficulty of recording ground truth trajectories in indoor settings, most indoor 3D reconstruction research is concentrated on a few datasets and in turn, a few sensors. Since the sensors considered in this work are not used in any already openly available dataset, we collect a new dataset and use three different evaluation approaches: comparison of the trajectory against traditional LiDAR systems, evaluation



Fig. 2. An example frame of each sequence captured in Experiment I.

of the the end-to-end error on sequences where the system goes back to the origin, and evaluation of estimated distances in the resulting point clouds against distances scanned with a traditional LiDAR. Additional information on the proposed dataset are available on the project’s webpage. All of our experiments are executed on a laptop with an 11th Gen Intel Core i9-11900H CPU. The Azure Kinect records depth maps and RGB images at 15FPS and the Intel Realsense at 30 FPS.

A. Experiment I: Large and small spaces, ablation study

We use DVI Mapper with the Microsoft Azure Kinect and evaluate against reference data acquired using FAST-LIO2-SLAM³ with a Livox Avia LiDAR, as shown in Figure 1. The sequences captured include trajectories starting from and navigating around a semi-indoor parking lot (*parking*), a large stair hall (*stairs*), a class room (*class*) and a meeting room (*meeting*), as shown in Figure 2. All sequences last approximately one minute, go through multiple turns and multiple rooms, and are recorded holding the sensor setup with one hand. We evaluate the following algorithms: VoxelMap (corresponding to the DOM component of DEVIL), VINS-Mono (the VIO component), and various ablated versions of our system, enabling or disabling DOM, VIO, the solution remapping component (RM) or the IMU propagation component (IMU). We have also experimented with SSL-SLAM [2] and ORB-SLAM3-RGBD [20], but neither of these algorithms successfully processed any of the four sequences without failing due to tracking loss.

TABLE II
EXPERIMENT I - ABSOLUTE POSE RMSE, REPORTED IN METERS AND DEGREES.

	<i>stairs</i>		<i>parking</i>	
	[m]	[deg]	[m]	[deg]
DOM (VoxelMap)	1.411	8.073	0.556	1.875
VIO (VINS-Mono)	0.799	4.154	0.481	0.688
DOM, RM	0.981	3.904	0.276	1.275
DOM, IMU	0.13	2.27	0.201	1.071
DOM, VIO, IMU	0.129	1.013	0.215	0.864
DOM, RM, IMU	0.147	2.181	0.195	0.726
DOM, RM, IMU, VIO	0.126	1.065	0.171	0.677

We first report the position and rotation RMSE in Table II for the *stairs* and *parking* sequences. (the *meeting* and *class* sequences are unsuitable for this reference setup, as the narrower geometry creates degeneracy for the Livox Avia). Then, we report the the end-to-end position error in Table III,

³A variant of FAST-LIO2 which implements loop closure: https://github.com/gisbi-kim/FAST_LIO_SLAM

TABLE III
EXPERIMENT I - END-TO-END POSITION ERROR, REPORTED IN METERS.

	<i>class</i>	<i>meeting</i>	<i>stairs</i>	<i>parking</i>
DOM (VoxelMap)	3.109	3.304	3.115	1.401
VIO (VINS-Mono)	0.425	2.123	3.758	2.664
DOM, RM	3.109	3.304	2.059	0.380
DOM, IMU	0.005	0.019	0.039	0.711
DOM, IMU, VIO	0.003	0.019	0.915	0.728
DOM, RM, IMU	0.005	0.019	0.299	0.580
DOM, RM, IMU, VIO	0.003	0.019	0.304	0.227

for all four sequences. DVI Mapper and its variants achieve a robust performance, especially compared to previous depth-base methods.

TABLE IV
EXPERIMENT I - TIME TO PROCESS ONE FRAME, IN MILLISECONDS.

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

On Table IV, we report the average time to process one frame in each DVI Mapper configuration. Without surprise, the two main sub-systems (DOM and VIO) take up the most processing time. Then, the IMU propagation and solution remapping both add only a few milliseconds of processing time. We specifically remark the performance increase thanks to solution remapping (RM), which comes at a very little computational cost. The complete system takes just below 100ms to process a frame, which means that it can process 10 FPS sequences in real-time.

B. Experiment II: Slow and fast motion

We reuse the same sensor system as in Experiment I and capture two scenes: a large utility room (*utility*) and a long cellar (*cellar*) with poor lighting and few geometric features. For each scene, we first record data following a slow trajectory (max 3m/s^2 and $30^\circ/\text{s}$) and then following a fast, shaky trajectory (up to 6m/s^2 and $100^\circ/\text{s}$). A picture

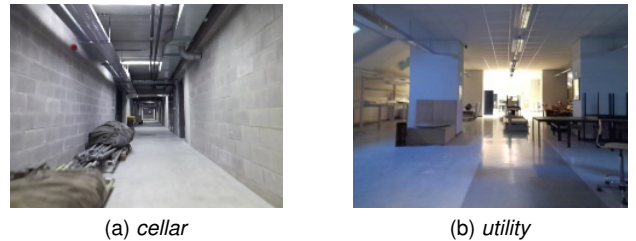


Fig. 3. Both scenes captured in the Experiment II.

of both scenes is shown in Figure 3. We run experiments with VoxelMap, VINS-Mono, SSL-SLAM and our proposed DVI Mapper. We report the average absolute position errors in Table II and plot the results in the utility room scene in

Figure 4. In all sequences, SSL-SLAM crashes before the end. While VoxelMap and VINS-Mono maintain correct tracking in the slow variant, their performance becomes inferior to DVI Mapper in the fast variant.

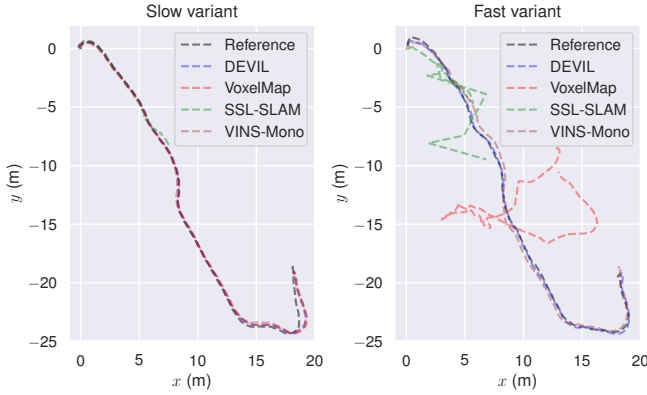


Fig. 4. Experiment II - utility. Our method achieves superior results, especially in the fast variant.

TABLE V
EXPERIMENT II - AVERAGE ABSOLUTE POSITION ERRORS IN METERS.

	Slow		Fast	
	<i>cellar</i>	<i>utility</i>	<i>cellar</i>	<i>utility</i>
VoxelMap	0.19	0.40	DNC	DNC
SSL-SLAM	DNC	DNC	DNC	DNC
VINS-Mono	0.14	0.30	0.92	0.40
DEVIL (ours)	0.18	0.27	0.17	0.21

Here, sequences where the algorithm tracking failed before the end of the sequence are indicated by DNC (Did Not Complete). VINS-Mono produces a better trajectory in one of the four sequences. This could be explained by the fact that the slow cellar sequence is particularly fitted for visual odometry (reliable visual features, especially with slow rotations) and not depth-based odometry (few salient geometrical features). On the three other sequences, DVI Mapper achieves the best performance.

C. Experiment III: Office and lab scenes with Realsense L515

We capture a dataset featuring a large loop in a lab environment (*lab loop*) and a long corridor with few visual and geometric features (*corridor*) using the system with Intel Realsense L515 as shown in Figure 1b. Both environments are captured with three different sensor setups: on a cart (slow and steady motion), hand-held slow (relatively steady) and hand-held fast (rapidly moving the sensor in rotation and translation). We perform an evaluation in two stages, starting with an qualitative evaluation of the global consistency of the point clouds. As an example, we display the results of the hand-held, slow sequences in Figure 5. SSL-SLAM crashes in the first few frames. VoxelMap does not crash but still fails to reproduce the full length of the corridor.

We then perform a quantitative evaluation using the distance between three point pairs in each environment, shown in

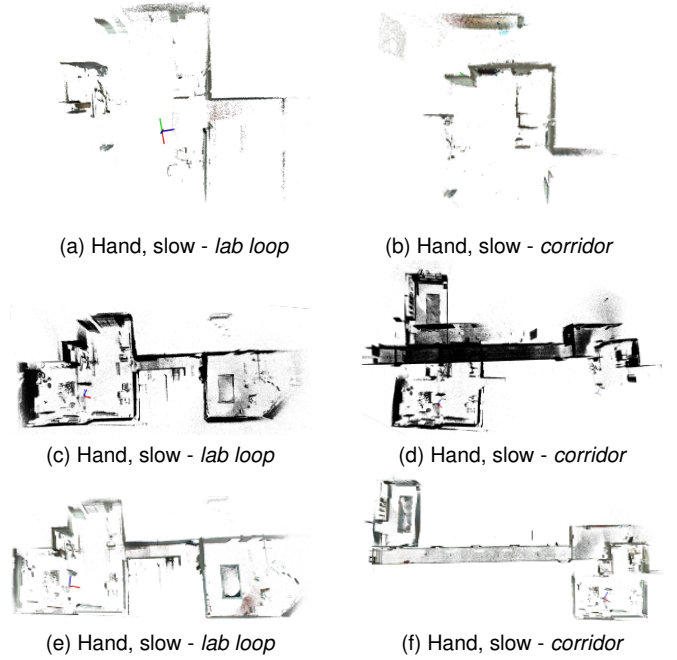


Fig. 5. Experiment III - Point clouds acquired with SSL-SLAM (top, all runs crashed before the end of the sequence), VoxelMap (middle), and our system.

TABLE VI
EXPERIMENT III - EVALUATION DISTANCES IN THE REFERENCE POINT CLOUDS, IN METERS.

<i>lab loop</i>			<i>corridor</i>		
S1	S2	S3	S1	S2	S3
25.89	9.81	6.74	29.72	12.17	9.4

Figure 6 and Table VI. These point pairs are chosen to represent long-range relations (observed by the sensor from poses that are far off each other). Since we evaluate the mapping output in this experiment, we cannot evaluate VINS-Mono or ORB-SLAM as these methods do not output a dense map. As it can be seen on the relative distance errors reported in Table VII, the results of Figure 5 are representative of the whole dataset: our method achieves a significantly more robust and precise performance. In this table, F means that the algorithm failed to produce a point cloud that allows to correctly find both ends of the segment. SSL-SLAM failed to do so in all sequences, so it is not reported in the table. We see that our method outperforms VoxelMap in 13 out of 15

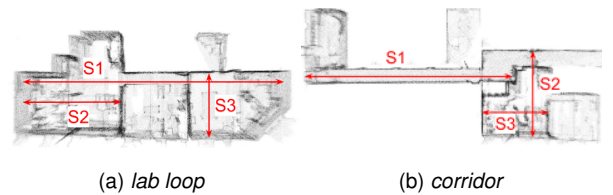


Fig. 6. Experiment III - Point clouds of the reference data, with the segments (S1, S2, S3) that we use for evaluation in both environments.

cases and often by a large margin.

TABLE VII
EXPERIMENT III - RELATIVE DISTANCE ERRORS IN %.

<i>lab loop</i>						
	S1		S2		S3	
	VoxelMap	ours	VoxelMap	ours	VoxelMap	ours
Cart	-3.86	-0.81	-3.05	-1.73	-5.19	-2.96
Hand, slow	-1.43	-0.46	-1.93	-3.26	-4.89	-4.45
Hand, fast	F	0.62	F	4.18	-3.26	-2.67

<i>corridor</i>						
	S1		S2		S3	
	VoxelMap	ours	VoxelMap	ours	VoxelMap	ours
Cart	-42.02	-3.36	F	1.40	F	2.44
Hand, slow	-15.78	1.65	F	-1.23	1.27	2.23
Hand, fast	F	F	F	F	F	F

V. 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 DVI sensors such as off-the-shelf depth cameras, 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 novel, challenging datasets. These experiments have shown the superior performance of our method against methods developed for similar sensors. To benefit the community, we open source both our code and datasets.

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/kineect-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] M. Vlaminc, 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.
- [9] M. Vlaminc, H. Luong, and W. Philips, "Liborg: a lidar-based robot for efficient 3D mapping," in *Applications of Digital Image Processing XL*, vol. 10396, no. 1. SPIE, 2017.
- [10] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.
- [11] 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.
- [12] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "Slit: 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.
- [13] 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. [Online]. Available: <https://api.semanticscholar.org/CorpusID:267412335>
- [14] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-lid: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4861–4868, 2022. [Online]. Available: <https://api.semanticscholar.org/CorpusID:247231064>
- [15] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," 2021.
- [16] W. Xu and F. Zhang, "Fast-lid: 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.
- [17] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lid2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.
- [18] J. Lin, C. Zheng, W. Xu, and F. Zhang, "R² 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.
- [19] 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. 10672–10678.
- [20] 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.
- [21] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "Deep learning for visual localization and mapping: A survey," *IEEE transactions on neural networks and learning systems*, vol. PP, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:261242607>
- [22] Hamesse, Charles and Vlaminc, Michiel and Luong, Hiep and Haelterman, Rob, "Practical Deep Feature-Based Visual-Inertial Odometry," in *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM)*. Scitepress, 2024, pp. 240–247.
- [23] I. Vasiljevic, V. Guizilini, R. Ambrus, S. Pillai, W. Burgard, G. Shakhnarovich, and A. Gaidon, "Neural ray surfaces for self-supervised learning of depth and ego-motion," in *International Conference on 3D Vision*, 2020.
- [24] 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.
- [25] P. Geneva, K. Eickenhoff, 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. [Online]. Available: <https://api.semanticscholar.org/CorpusID:204745914>
- [26] 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. [Online]. Available: <https://api.semanticscholar.org/CorpusID:119105448>
- [27] C. Hamesse, H. Luong, and R. Haelterman, "Evaluating the impact of head motion on monocular visual odometry with synthetic data," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, vol. 5, Farinella, G. M. and Radeva, P. and Bouatouch, K., Ed. Scitepress, 2022, pp. 836–843.
- [28] J. Mo and J. Sattar, "Continuous-time spline visual-inertial odometry," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9492–9498, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:237571712>
- [29] J. Solà, "Quaternion kinematics for the error-state kf," 2015.
- [30] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022. [Online]. Available: <https://github.com/ceres-solver/ceres-solver>
- [31] 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.