Digital Analytics and Robotics for Sustainable Forestry

CL4-2021-DIGITAL-EMERGING-01
Grant agreement no: 101070405

DELIVERABLE 3.2

Field results: Per-platform multi-modal, state estimation software stacks

Due date: month 24 (August 2024)
Deliverable type: R
Lead beneficiary: TUM

Dissemination Level: PUBLIC

Contents

1 Introduction

State estimation plays a crucial role in field autonomy by providing robust and accurate robot states in real-time. In this deliverable, the objective is to present platform-specific state estimation software stacks employed in aerial robots, the ANYmal legged robot, and the Supervised Autonomous Harvest (SAHA) robot. Specifically, the aerial robots include NTNU Resilient Micro Flyer (RMF), Leica BLK2FLY, and TUM Smart Robotics Lab (SRL) drone. Quantitative as well as qualitative localization and mapping results are shown that serve as a baseline of the state estimators in the DigiForest project. Experimental results include both the field trials and benchmark datasets where we compare presented state estimators to state-of-the-art methods.

This document is organized by robot platforms, and each chapter is organized by following three Tasks:

- Multi-modal state estimation,
- Detection and handling of sensor degradation and failure,
- Long-term and viewpoint tolerant place recognition and localization.

The state estimators are multi-modal in the sense that different types of sensors, which include interoceptive (IMUs, joint encoders) and exteroceptive sensors (LiDARs, RGBD sensors), are fused for maximum robustness and accuracy. To further push resilience, the state estimator detects sensor degradation and failure and discards the corresponding measurements in the SLAM/odometry factor graph optimisation. Furthermore, place recognition and localization enable long-term loop closure under different sensor modalities and substantial view-point changes.

Notations Throughout this deliverable, the following notation will be used: coordinate frames are written as $\underrightarrow{\mathcal{F}}_A$ and a vector expressed in this reference frame will be denoted as ${}_A\mathbf{r}$. The rigid body transformation which transforms points from a reference frame $\underrightarrow{\mathcal{F}}_B$ to another reference frame $\underrightarrow{\mathcal{F}}_A$ is given by $\boldsymbol{T}_{AB} \in SE(3)$ and can be decomposed into a rotation matrix $\mathbf{C}_{AB} \in SO(3)$ and the translational component ${}_A\mathbf{r}_B$. We also denote the rotation $\mathbf{C}_{AB}$ with its unit quaternion form $\mathbf{q}_{AB}$. The most important reference frames that will be used are: a fixed world reference frame $\underrightarrow{\mathcal{F}}_W$, the camera coordinate frames $\underrightarrow{\mathcal{F}}_{C_i}$ for $i = 1 \dots N$ cameras, the IMU sensor frame $\underrightarrow{\mathcal{F}}_S$, the LiDAR sensor frame $\underrightarrow{\mathcal{F}}_L$ and a map frame $\underrightarrow{\mathcal{F}}_M$. Furthermore, $[\cdot]^\times$ represents the skew-symmetric matrix of a 3D vector. If these definitions are not used, new notations will be defined in the corresponding sections.

2 Aerial robots

The task of aerial robots in the DigiForest project is an autonomous exploration in the forest to record multi-modal sensor measurements including point clouds and images and to map the environments including individual trees. Aerial robots serve as a crucial entity in this project since they are not hindered by traversability and can efficiently scan the environment. The collected data is further utilized to build a highly precise forest model to support decision-making. To achieve this, there are three aerial robot platforms that are equipped with cameras, inertial measurement unit (IMU), and optionally LiDAR.

2.1 Resilient Micro flyer (RMF)

2.1.1 Multi-modal state estimation

Multi-modal sensors The RMF-Owl is a collision-tolerant drone built by ARL at NTNU, and shared with SRL at TUM. It is a $\approx 1.5kg$ platform embarking an Ouster OS0-64 [28] LiDAR, a Flir Blackfly S color camera, and a VN100 IMU for localization and mapping. The LiDAR has a $360 \times 90^{\circ}$ FoV and provides scans at 10 or 20 Hz. The camera has a FoV of $85 \times 64^{\circ}$ and has a synchronized global shutter, providing images at 30 Hz.

State estimation The state estimator is based on CompSLAM [17], which relies on the LiDAR odometry and mapping concept introduced in [47]. It performs loose coupling of the sensor data. The LiDAR odometry consist of estimating the transform between successive LiDAR scans via alignment of geometric features. It makes use of point-to-line and point-to-plane metrics, as those demonstrated good robustness to large-scale localization [47], scalability to the high density pointclouds without performing computationally expensive point-to-point matching. Mapping is achieved by registering new LiDAR scans to an aggregated map, performing scan-to-map matching.

The compSLAM algorithm exploits, in addition to LiDAR, additional localization modalities provided by, e.g., Visual Inertial Odometry (VIO). The VIO is merged in a loosely-coupled fashion, ensuring continuity of observations when the geometric features are not sufficient for ensuring a reliable estimation. This is typically the case in self-similar environments, or in very sparse environment where scan-to-scan matching becomes computationally expensive, e.g. in open forest environments. Camera odometry estimates are obtained at a higher rate than pointcloud estimates. Hence, we use camera odometry to estimate the relative transformation between the successive pointclouds, and provide it as a prior for the scan-to-scan matching, therefore improving the convergence rate of the pointcloud alignment process. The transformation prior is used to predict the position of points at the current position. The residual error, i.e. the sum of squared distances between the predicted and the measured points, is used as the cost function to be minimized. Additionally, analyzing the eigenvalues of the Jacobian of the aforementioned cost function provides insights on failure of the underlying optimization process, and thus on the failure of the LiDAR scan matching. In such an event, the VIO transform is used to propagate the LiDAR odometry and mapping. More details on sensor failure detections are provided in Section 2.1.2. The algorithm outline is presented in the block diagram in Fig. 1.

Finally, the LiDAR odometry, computed at 5 Hz, is fed to an EKF [24] fusing IMU measurements for interpolating the odometry at the controller frequency, i.e. 100 Hz.

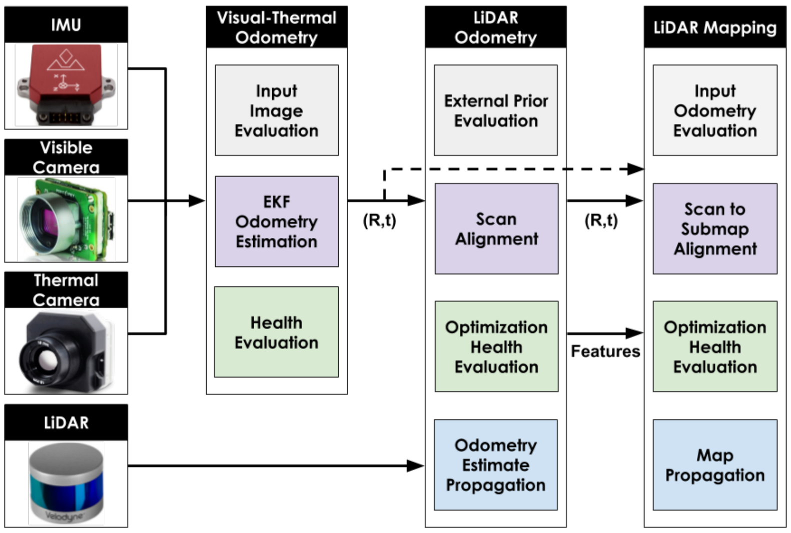


Figure 1: Block diagram of the CompSLAM algorithm.

Experiments The state estimator was used onboard the drones for exploration missions, during the field testing campaigns in Evo (May 2023) and in Stein am Rhein (July 2024). The Ouster scans are processed to estimate the robot state at create a pointcloud map of the environment. The state estimation was qualitatively evaluated in various environments with various tree densities and topography. Fig. 2 and 3 respectively illustrate pointcloud maps obtained in Evo and Steim am Rhein. The RVIZ visualization depicts the registered maps throughout the mission via CompSLAM running onboard, at 5Hz. The two main challenges faced regarding the localization were

1. when going too high and thus loosing visibility of the ground close to the robot, the scan-to-map registration was taking longer than usual due to the lack of planar geometric features, resulting in a slightly reduced LiDAR odometry frequency;

2. when moving at significant speed, the LiDAR scans were subject to a significant skewing (due to the drone displacement during a data acquisition step). This issue is greatly alleviated by increasing the Ouster frequency to 20 Hz, therefore trading off the pointcloud quality for a reduced skewing. In this setup, no odometry breakage was recorded in the various test flights in the forest.

The results qualitatively offer stable odometry both in open and cluttered environments. Centimetric-level drift was achieved in the 170 m long mission (back and forth) depicted in Fig. 2.

2.1.2 Detection and handling of sensor degradation and failure

CompSLAM detects LiDAR degeneracy through a threshold check on the eigenvalues of the underlying Jacobian of the utilized pseudo-Hessian. In the nominal implementation, such threshold is tuned manually but experience has shown that a threshold is enough to be tuned for a class of environments and there is no need for environment-specific tuning. Details are provided in [17].

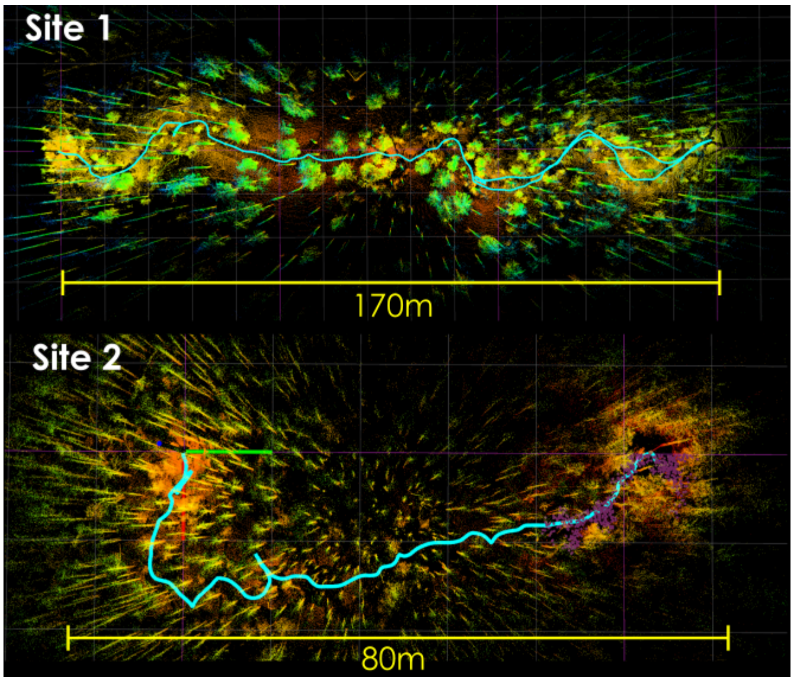


Figure 2: Pointcloud maps obtained from two exploration mission with RMF-Owl, in Evo, Finland. The mapping is performed in two sites of different tree densities, relatively sparse for the top image and very cluttered for the bottom one.

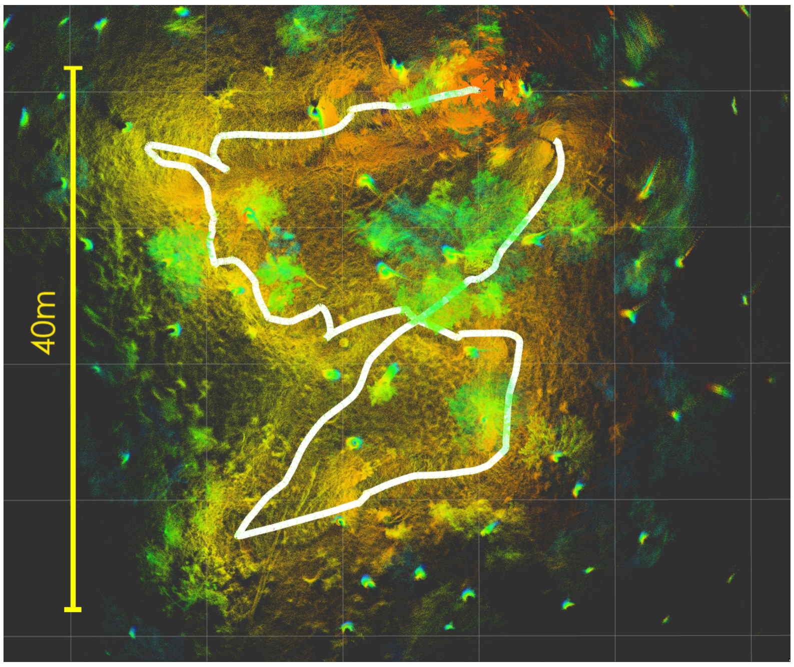


Figure 3: Pointcloud maps obtained from an exploration mission of a 40mx40m area, in Stein am Rhein, Switzerland.

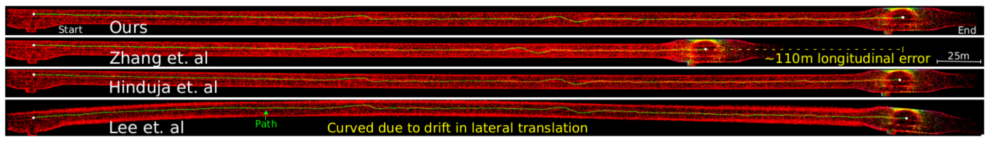


Figure 4: Partial maps using our new method and also three others in the Fyllingsalen bicycle tunnel at Bergen Norway. As shown all other methods make wrong estimation of scale and drift rotationally.

When degeneracy is detected in LiDAR odometry and mapping, CompSLAM passes through its visual-inertial estimate. As described in the abovementioned work it is an assumption that at any given time at least LiDAR registration or visual-inertial odometry estimation is well-posed (i.e., their data are not degraded and thus the problem not ill-conditioned).

Nevertheless, it is still a fact that for different sensors or for substaintially different environments heuristic tuning of such thresholds is necessary. Responding to these limitations we have recently further looked into a more comprehensive treatment. Specifically, we have focused on a tuning-free approach to address degeneracy-aware LiDAR localization and mapping with attention on detecting uninformative conditions in Iterative Closest Point (ICP) solving as they manifest due to geometries not offering constraints across all degrees of freedom.

Generally, the Hessian matrix represents 2nd order information in the optimization process and degeneracy occurs when certain directions in the matrix have very low eigenvalues. Existing methods put hand-tuned thresholds. In our new work we model signal-to-noise ratio in each direction allowing us to seamlessly identify degenerate directions by simply calculating if the signal exceeds the noise by an order of magnitude (no particular sensitive tuning).

Experiments We have verified the benefits of this method in an environment where degeneracy direction is obvious and persistent, namely in a perfectly self-similar straight long tunnel and compared against other methods of the state-of-the-art as presented in Fig. 4. In this result we also compared against the works presented in [46, 12, 22].

2.1.3 Place recognition and localization

Within its nominal operation, CompSLAM retains a “registered cloud” which essentially represents the collective representation of the environment so far – as estimated at every step of point cloud registration – albeit sparsified for purposes of computational and memory efficiency. Sparsification is achieved by applying a voxel filter on the cloud. In the default settings, the registered cloud represents a global map representation with maximum dimensions of $500 \times 500 \times 500$m, a value that can be modified and depends on the computational resources available (the value mentioned represents the one used by our RMF flying robots).

This registered cloud is used at every mapping step of CompSLAM in that the current point cloud is registered not only against the previous (odometry-like optimization) but also against the maintained “global map”, i.e. the registered cloud.

At the same time, CompSLAM offers the possibility to load at the beginning of a mission a point cloud file which then initializes the registered cloud. This allows either to a) use a map from a previous mission (from any robot) to localize against the older map, and b) send such a map to another robot within the same mission (for example when ANYmal deploys the marsupially-ferried RMF as demonstrated by our NTNU team).

2.2 BLK2FLY

2.2.1 Multi-modal state estimation

Multi-modal sensors BLK2FLY by Leica is equipped with a single-beam dual-axis spinning LiDAR with 360° vertical and 270° horizontal field of view (FoV), 5 global shutter cameras with 300° horizontal and 180° FoV in total, and 5 IMUs. The LiDAR scan ranges from 0.5m to 25m and delivers 420,000 points/sec. The multi-modal measurements from the LiDAR, cameras, and a single IMU are fed to the state estimator.

State estimation The state estimator is based on OKVIS2 [23], its extended work to fuse lidar measurements [2], and adaptive resolution dense mapping library Supereight2 [11]. The state vector is defined as

$$\mathbf{x} = \begin{bmatrix} {}_W\mathbf{r}_S^T & \mathbf{q}_{WS}{}^T & {}_W\mathbf{v}^T & \mathbf{b}_\mathrm{g}^T & \mathbf{b}_\mathrm{a}^T \end{bmatrix}^T, \tag{1}$$

where ${}_W\mathbf{r}_S$ and $\mathbf{q}_{WS}$ denote the position and orientation of the IMU sensor frame in the fixed world frame, and ${}_W\mathbf{v}$ describes the velocity of the IMU with respect to the world frame. $\mathbf{b}_\mathrm{g}$ and $\mathbf{b}_\mathrm{a}$ stand for gyroscope and accelerometer biases, respectively.

The factor graph representation used in this work is shown in Fig. 5. On top of visual, inertial and pose graph factors, that are already used in OKVIS2, two different types of LiDAR factors are added to the graph. For every state in the real-time optimization window, LiDAR factors with respect to the last completed submap are added. Furthermore, map-to-map LiDAR factors are added to constrain submaps with respect to each other by aggregating measurements between submaps.

Specifically, OKVIS2 [23] uses 2D reprojection errors $\mathbf{e}_\mathrm{r}^{i,j,k}$ of the j-th landmark in the frame of the i-th camera at a timestamp k and its corresponding observation $\tilde{\mathbf{z}}_\mathrm{r}^{i,j,k}$ in the image is given by

$$\mathbf{e}_\mathrm{r}^{i,j,k} = \tilde{\mathbf{z}}_\mathrm{r}^{i,j,k} - \mathbf{h}\left(\boldsymbol{T}_{SC_i}^{-1}\boldsymbol{T}_{S^kW}\,{}_W\mathbf{l}^j\right). \tag{2}$$

Hereby, $\mathbf{h}(\cdot)$ denotes the camera projection. For the formulation of the IMU residuals, OKVIS2 adopts the IMU preintegration approach in [9]. Between time steps k and n, the IMU error is:

$$\mathbf{e}_\mathrm{s}^k = \tilde{\mathbf{x}}^n\left(\mathbf{x}^k, \tilde{\mathbf{z}}_\mathrm{s}^{k,n}\right) \boxminus \mathbf{x}^n, \tag{3}$$

where $\tilde{\mathbf{x}}^n$ is the predicted state at an arbitrary time n as a function of the current state $\mathbf{x}^k$ and IMU measurements $\tilde{\mathbf{z}}_\mathrm{s}^{k,n}$. The $\boxminus$ performs regular subtraction except for the quaternion (see [23]). Additionally, relative pose errors $\mathbf{e}_\mathrm{p}^{r,c}$ between time steps r and c are given by

$$\mathbf{e}_\mathrm{p}^{r,c} = \mathbf{e}_{\mathrm{p},0}^{r,c} + \begin{bmatrix} {}_{S^r}\mathbf{r}_{S^c} - {}_{S^r}\tilde{\mathbf{r}}_{S^c} \\ \mathbf{q}_{S^rS^c} \boxminus \tilde{\mathbf{q}}_{S^rS^c} \end{bmatrix}. \tag{4}$$

with ${}_{S^r}\tilde{\mathbf{r}}_{S^c}$ and $\tilde{\mathbf{q}}_{S^rS^c}$ being nominal relative position and orientations expressed in the IMU frame $\underrightarrow{\mathcal{F}}_S$. We refer the reader to [23] for a detailed derivation of the constant $\mathbf{e}_{\mathrm{p},0}^{r,c}$ as well as error Jacobians and error weights $\mathbf{W}_\mathrm{p}^{r,c}$.

There are two types of LiDAR residuals. In both cases, the error residuals are formulated based on two states. On the one hand, for live residuals, these two states

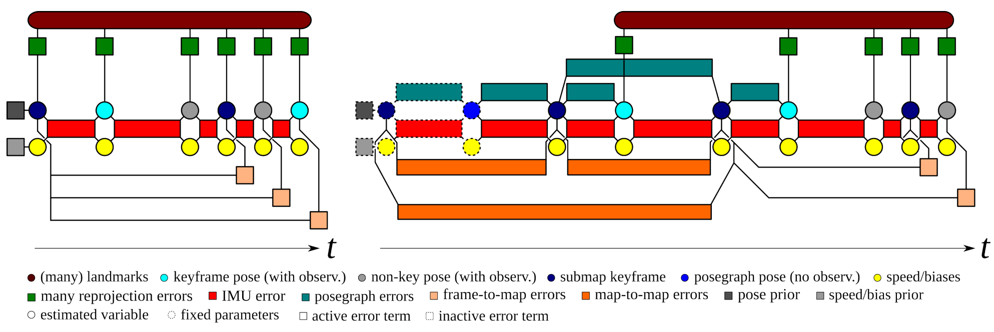


Figure 5: Optimization Factor Graph of the LiDAR-Visual-Inertial Estimator. Left: The real-time estimator connects set of current keyframe states and non-keyframe states by IMU errors and visual reprojection errors. It also shows the active submap keyframe and the last completed submap keyframe. For every state in the optimization window, we can formulate live LiDAR factors between every live frame and the last completed submap. Right: OKVIS2 connects keyframe poses through relative pose errors at a later stage. In addition to the live LiDAR factors, measurements between frames can be aggregated and map-to-map LiDAR factors can be added to the factor graph. Every map will be connected to the previous submap and optionally an older submap if the geometric overlap surpasses a threshold.

are given by the current frame and the keyframe associated to the last completed submap. On the other hand, for map-to-map residuals, both states will be submap anchor frames. Regarding the formulation of the error residuals, however, they do not differ. Given a completed submap anchored at keyframe pose $\boldsymbol{T}_{WS^a}$ and a point cloud $\mathcal{P}_b$ associated to another state $\boldsymbol{T}_{WS^b}$, we formulate the residual for every ${}_{S^b}\mathbf{p} \in \mathcal{P}_b$ as:

$$e_{\mathrm{l}}^{a,b}\left({}_{S^a}\mathbf{p}\right) = \frac{d}{\sigma} = \frac{L\left({}_{S^a}\mathbf{p}\right)}{\sqrt{\frac{L_{\mathrm{min}}^2}{9} + \sigma_z^2\left|\nabla L\left({}_{S^a}\mathbf{p}\right)\right|^2}}, \tag{5}$$

where L is a occupancy log-odd, ∇L is the gradient of L, L_{min} is a configurable parameter for the minimum log-odd, and ${}_{S^a}\mathbf{p} = \boldsymbol{T}_{S^aS^b}\,{}_{S^b}\mathbf{p}$ with $\boldsymbol{T}_{S^aS^b} = \boldsymbol{T}_{WS^a}{}^{-1}\boldsymbol{T}_{WS^b}$.

All of the aforementioned factors are combined in the overall minimisation objective:

$$\begin{aligned} c(\mathbf{x}) = &\frac{1}{2}\sum_{i}\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}(i,k)} \rho\left(\mathbf{e}_{\mathrm{r}}^{i,j,k\,T}\mathbf{W}_{\mathrm{r}}\mathbf{e}_{\mathrm{r}}^{i,j,k}\right) \\ &+ \frac{1}{2}\sum_{k\in\mathcal{P}\cup\mathcal{K}\setminus f} \mathbf{e}_{\mathrm{s}}^{k\,T}\mathbf{W}_{\mathrm{s}}^{k}\mathbf{e}_{\mathrm{s}}^{k} + \frac{1}{2}\sum_{r\in\mathcal{P}}\sum_{c\in\mathcal{C}(r)} \mathbf{e}_{\mathrm{p}}^{r,c\,T}\mathbf{W}_{\mathrm{r}}^{r,c}\mathbf{e}_{\mathrm{p}}^{r,c} \\ &+ \frac{1}{2}\sum_{k\in\mathcal{K}}\sum_{\mathbf{p}\in\mathcal{L}_k} e_{\mathrm{l}}^{C,k\,2} + \frac{1}{2}\sum_{b\in\mathcal{M}}\sum_{a\in\mathcal{A}_b}\sum_{\mathbf{p}\in\mathcal{L}_b} e_{\mathrm{l}}^{a,b\,2}. \end{aligned} \tag{6}$$

Here, the set $\mathcal{K}$ contains the most recent frames as well as keyframes with observations of visible landmarks in $\mathcal{J}(i,k)$. $\mathcal{P}$ contains all pose graph frames and f denotes the most current frame. $\mathcal{C}(r) \subset \mathcal{P}$ is the set of all pose graph frames connected to a frame r. Furthermore, the set $\mathcal{L}_k$ denotes the set of all LiDAR measurements associated to a frame k. $\mathcal{M}$ is the set of all past submaps, and $\mathcal{A}_b$ the set of all submap frames connected to a submap frame $\boldsymbol{T}_{WS^b}$ via map-to-map residuals. C denotes the last completed submap frame.

Experiments The presented estimator is quantitatively evaluated on the HILTI 2022 SLAM Challenge benchmark [48] where the sensor configuration is similar to

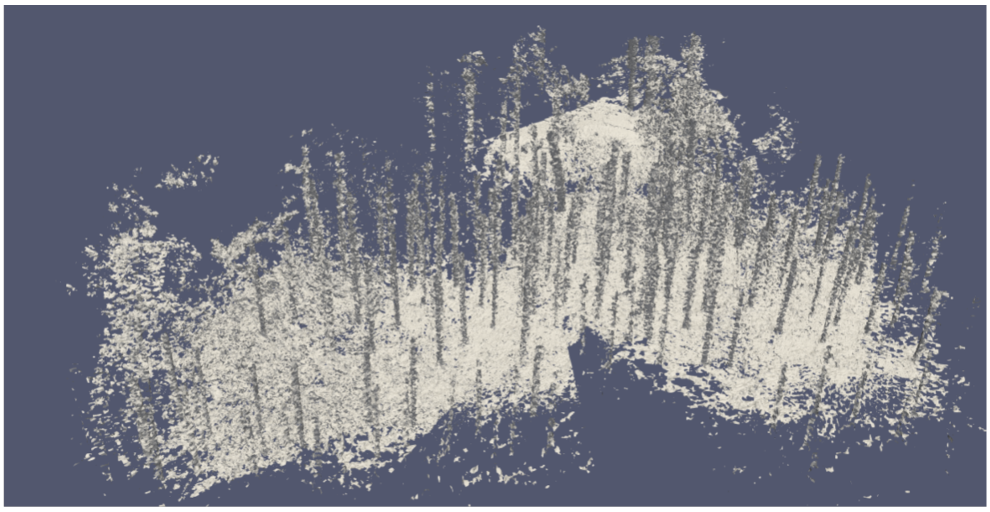


Figure 6: 3D mesh reconstructed by BLK2FLY in the field trip, Stein am Rhein (July 2024).

Approach		**Sensors**			**Sequence**								**Score**
	C	L	V	I	exp01	exp02	exp03	exp07	exp09	exp11	exp15	exp21	
OKVIS2 [23]	✓		✓	✓	8.46	8.18	0.00	0.00	0.00	0.00	0.00	0.00	16.64
OKVIS2 [23]			✓	✓	30.77	20.00	0.00	31.67	13.75	28.00	12.22	0.00	136.41
VILENS [39]		✓	✓	✓	69.23	49.09	40.00	16.67	5.00	68.00	17.78	60.00	325.77
HKU*	✓	✓	✓	✓	**84.62**	70.00	44.71	31.67	7.50	**92.00**	24.44	48.00	402.93
Wildcat [33]		✓		✓	**84.62**	**84.55**	**90.59**	**45.00**	**44.38**	84.00	46.67	**84.00**	**563.79**
Ours	✓	✓	✓	✓	43.08	17.27	32.35	0.00	16.25	32.00	51.11	62.00	254.06
Ours		✓	✓	✓	62.31	39.09	42.35	13.33	38.12	**92.00**	**64.44**	76.00	427.65

Table 1: Evaluation scores on HILTI22 SLAM Challenge. Bold: best score, underlined: second best; C: causal evaluation (definition in [23]); L, V, I: LiDAR, Visual and Inertial measurements are considered; *based on [49].

that of BLK2FLY. The Hilti-Oxford dataset was collected using a handheld sensor device consisting of an Alphasense five-camera module, an IMU and a 32 beam Hesai PandarXT-32 LiDAR sensor.

For evaluating the position accuracy, sparse ground-truth with millimeter accuracy is provided. An automatic evaluation system is available that uses a score based on the Absolute Trajectory Error (ATE) as an evaluation metric. First, the estimated trajectories are aligned with the sparse ground-truth using $SE(3)$ Umeyama alignment. For every ground-truth control point, the corresponding estimate is retrieved and awarded a score ranging from 0 to 10 (10 for errors below 1 cm, 0 for errors above 10 cm, for more details see [48]).

Table 1 shows the final scores of our approach on all challenge sequences with identical parameters. We report the causal as well as non-causal estimates including a final Bundle Adjustment. We compare it to the Visual-Inertial SLAM (including loop closures) performance of OKVIS2 [23] as a baseline. The VI baseline used all 5 cameras and the same visual frontend parameters as in the LVI case. Furthermore, we compare it to other state-of-the-art methods that have been published an evaluated on the benchmark including the – at the time of the challenge – winning algorithm Wildcat [33]. It can be seen that adding LiDAR factors increases the accuracy of OKVIS2 significantly, in the causal as well as the final optimised evaluation. Furthermore, the proposed approach achieves state-of-the-art localization accuracy. At the moment, a final score of 427.65 achieves rank 7 out of 53 submissions on the challenge leaderboard. Note that also VILENS and Wildcat performed some form of offline batch or posegraph optimization for the reported scores. For a better understanding, it can be stated that the reported scores here denote a mean ATE of 1 cm (or even below) to 3 cm in the final estimates. Only for *exp07* where a long corridor with a lack of characteristic visual features or geometric features does not achieve the same performance and results in a mean ATE of approximately 7 cm.

The presented state estimator was also tested during the field trip in Stein am Rhein (July 2024). Onboard sensor measurements from BLK2FLY are streamed to a laptop, and the state estimator processes the streamed measurements in real-time. Fig. 6 qualitatively shows the 3D mesh that was reconstructed online by BLK2FLY.

2.2.2 Detection and handling of sensor degradation and failure

The state estimator in BLK2FLY fuses multi-modal sensor measurements including IMU, cameras, and LiDAR measurements. The estimator estimator probabilistically estimates the state by minimizing the optimization loss (6), which is a negative log-likelihood of the probability density functions. The estimator is robust to sensor failure in the sense that the optimization loss (6) is weighted by the inverse of measurement covariance matrices. For instance, the LiDAR factor (5) is down-weighted, if the log-odd gradient of the occupancy is low. Likewise, the weight for the IMU preintegration decreases with lower-grade IMUs. This naturally maintains proper weights among multi-modal sensors and discards measurements with high uncertainty.

The navigation stack relies on the state estimator and the mapping system which leverages log-odds to represent the scene. Thanks to this probabilistic representation, the planner can distinguish between occupied, free and unknown, with the latter representing unmapped areas. This representation allows us to select planned paths which only traverse what has been mapped as free space. This is an important safety mechanism since classical stereo-matching algorithms can fail in outdoor scenarios and only yield sparse depth output.

At the same time, drift is inevitable in state estimation due to sensor noise propagating to the optimization, leading to noisily estimated poses. This noise accumulates over time, leading to large drifts over long missions. The state estimator community has developed drift-correcting mechanisms to address this issue, with loop-closures being the default choice. Nonetheless, this drift correction poses a problem for navigation since sudden "jumps" in the estimated odometry can lead to potential crashes if the tracked trajectory is not updated correctly. To address this issue, we propose a method called "trajectory anchoring", which elastically deforms planned reference trajectories based on the state updates from the estimator.

Our approach, as visualized in Fig. 7, anchors each reference state to a set $\mathcal{S}$ of states $\mathbf{x}_s$, $s \in \{1, \dots, S\}$ estimated by OKVIS2. The reference trajectory states will be updated as the anchor states $\mathbf{x}_s$ update their corresponding poses. A reference trajectory contains references (superscript “ref”) $\mathbf{x}_j^{\text{ref}} = ({}_W\mathbf{r}_{WS_j}^{\text{ref}}, \mathbf{q}_{WS_j}^{\text{ref}}, {}_W\mathbf{v}_j^{\text{ref}})$, with $j \in \{1, \dots, J\}$ (and corresponding timestamps), which are anchored to the estimated OKVIS2 states in $\mathcal{S}$ via the algorithm of k-nearest neighbours, using the distance metric:

$$d_{j,s} = ||{}_W\mathbf{r}_{WS_j}^{\text{ref}} - {}_W\mathbf{r}_{S_s}||. \tag{7}$$

We associate every pair j, s with the relative transformation $\boldsymbol{T}_{S_sS_j}$ and a weight $w_{j,s}$:

$$w_{j,s} = \frac{1/d_{j,s}}{\sum_{j=1}^{S} 1/d_{j,s}}. \tag{8}$$

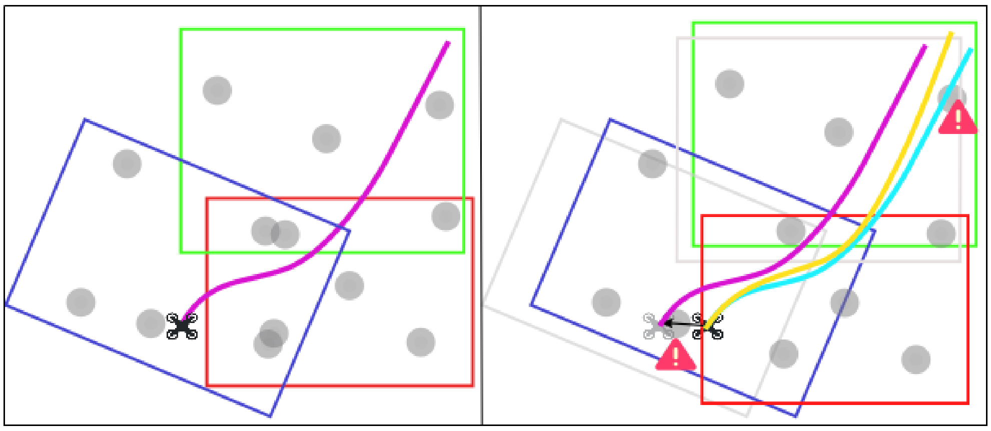


Figure 7: Planned trajectory before loop-closure (left) and trajectory adaptation strategies after loop-closure (right), with obstacles as gray disks. The adaptation strategies are: no adaptation (purple), rigid transformation (cyan) and the proposed trajectory anchoring (yellow). Without adaptation, the MAV flies towards its estimated pose before loop-closure (black arrow), which is unsafe. Rigidly transforming the trajectory does not account for the accumulated drift of each submap along it, leading to unsafe trajectories.

On state update, trajectory state positions are updated with:

$$ {}_W\mathbf{r}^{\text{ref'}}_{WS_j} = \sum_{s=1}^{S} w_{j,s}\, \boldsymbol{T}_{WS_s}\, {}_{S_s}\mathbf{r}_{S_j}\,, \tag{9} $$

where $\boldsymbol{T}_{WS_s}$ is the latest anchor pose as estimated by OKVIS2. The orientation $\mathbf{q}^{\text{ref'}}_{WS_j}$ is updated to the weighted average of $\mathbf{q}_{S_sS_j}$, $s \in \{1 \ldots S\}$ using the previously computed weights by following the method from [25]. We also apply the respective orientation changes to the velocities.

2.2.3 Place recognition and localization

A common challenge in robotics is to re-localize a robot within a given map. In our setting, this problem is especially challenging as we deploy two robot drones with different sensor modalities: The *BLK2FLY*, which we use to capture large-scale, sparse, LiDAR point clouds, and the *SRL drone*, which is equipped with a stereo camera and shall re-localize with in the BLK2FLY's maps.

One common approach in related work is to transform the point clouds into range images by rendering them from a specific perspective. However, this approach comes with two drawbacks: Firstly, by pointing the virtual camera into one direction only, one loses much of the point cloud's information and sacrifices a key advantage of the LiDAR sensor. Furthermore, projecting the 3d geometry into a 2d image loses much of the rich information provided by the LiDAR. An alternative method is to generate a 3D representation from the stereo image stream and subsequently register this representation with existing LiDAR maps. This approach, showcased in SOLVR [19], achieves state-of-the-art performance in LiDAR-Visual re-localization, advancing the robustness and accuracy of multi-modal registration.

The proposed method focuses on generating local 3D maps—so called *submaps*—from the stereo image stream and registering it against a database of LiDAR point clouds (Fig. 8). The submaps are generated using the same strategies as for the state estimator, making SOLVR a streamlined approach cross-modal place recognition with the generated data during missions with either LiDAR and stereo cameras to obtain depth input. To extract metric depth information from the stereo images, SOLVR utilizes the learning-based depth prediction network Unimatch [43]. The camera's odometry, generated through the visual-inertial odometry and a SLAM system OKVIS2 [23], provides the necessary trajectory for projection. A probabilistic occupancy mapping system, supereight2 [36], is utilized to mitigate the influence of noise and outliers by employing Bayesian updates to integrate depth measurements.

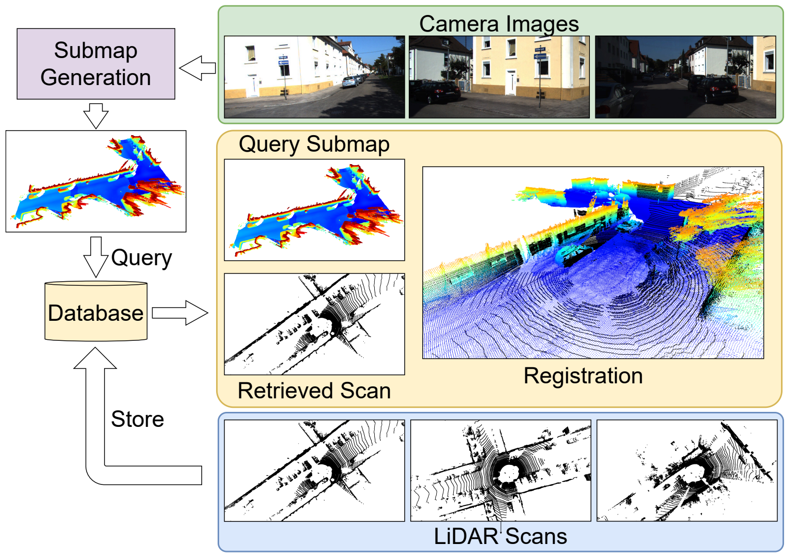


Figure 8: SOLVR constructs 3D submaps from a stream of input camera images in order to align the camera and LiDAR sensor modalities. The submap is used to retrieve a corresponding place from a database of LiDAR scans, at which point the submap and scan are registered in order to estimate the current sensor pose.

The query process comprises two steps: Querying the stereo-image-based submap in the database for a matching LiDAR point cloud (*place recognition*) and finding the relative pose between the two maps (*registration*). To perform place recognition, SOLVR extracts global feature embeddings from both 3D maps, which are 256-dimensional vectors generated by two sparse convolutional network, each tuned for one sensor modality. SOLVR retrieves the most similar place from the LiDAR database by comparing the global feature vector of the queried submap with the stored LiDAR-based embeddings.

For registration, SOLVR uses a secondary sparse convolutional network to extract local keypoint features and their corresponding 3D coordinates from both submap and LiDAR scan. These local features are then matched between the two maps to find correspondences. To deal with outliers during matching, SOLVR introduces a geometric consistency-based approach, which employs the first eigenvector of a feature correspondence matrix to weight contributions of each correspondence. Using these weights and the keypoint coordinates, the final 6D pose is computed using least-squares optimization. The paper shows that with this geometric consistency check, registrations are more successful—especially for low inlier-ratios—than with traditional RANSAC (Tab. 3).

SOLVR was evaluated on the KITTI and KITTI360 datasets. In LiDAR-Visual *place recognition*, SOLVR outperformed all state-of-the-art methods ScanContext [18],

Method	KITTI-00*				KITTI360-00				KITTI360-09			
	5 m		20 m		5 m		20 m		5 m		20 m	
	R@1↑	R@5↑	R@1↑	R@5↑	R@1↑	R@5↑	R@1↑	R@5↑	R@1↑	R@5↑	R@1↑	R@5↑
ScanContext[†] [18]	7.6	8.0	7.7	9.4	7.5	7.9	9.4	10.5	9.0	9.8	10.0	11.5
EgoNN[†] [20]	49.1	60.0	54.8	68.7	40.2	64.4	73.2	84.2	47.9	64.3	58.7	70.7
LIPLoc [35]	46.6	74.3	52.1	76.5	1.94	5.8	11.0	21.0	21.7	33.4	27.0	41.6
C2L-PR [44]	71.3	84.2	78.0	88.4	–	–	–	–	–	–	–	–
SOLVR (Ours)	**87.9**	**94.0**	**96.4**	**96.8**	**58.1**	**74.8**	**76.3**	**84.6**	**70.2**	**85.5**	**82.2**	**90.3**

Table 2: LiDAR-Visual Place Recognition results on the KITTI and KITTI360 datasets. [†] Indicates methods originally proposed for unimodal LiDAR tasks that have been adapted to LiDAR-Visual by using our generated submaps as queries. * Indicates sequences where the OKVIS2 [23] odometry was used to calculate the camera trajectory for submap construction.

Method	KITTI-09*			KITTI-10*		
	Acc.%	RRE (°)	RTE (m)	Acc.%	RRE (°)	RTE (m)
CorrI2P [34]	8.1	13.59	11.11	9.0	9.44	10.15
VP2P [50]	6.9	11.35	10.43	7.6	8.16	10.00
EgoNN[†] [20]	50.7	22.30	7.35	57.1	27.86	6.59
SOLVR-RANSAC	81.1	11.77	3.46	80.1	13.83	3.46
SOLVR (Ours)	**98.0**	**2.12**	**0.74**	**95.4**	**3.60**	**0.98**

Table 3: 6-DoF LiDAR-Visual registration results for the *Comprehensive* evaluation setup on KITTI-09 and 10

EgoNN [20], LIPLoc [35], and C2L-PR [44] as can be seen in Tab. 2.

Moreover, SOLVR's use of geometric consistency matrices for keypoint matching improved registration accuracy manifests in superior *registration* performance compared to the baseline methods CorrI2P [34], VP2P [50], and EgoNN (Tab. 3). For example, in the comprehensive evaluation setup, SOLVR demonstrated an accuracy of 98 % for 6-DoF registration on KITTI-00 compared to 81.1 % of the next best baseline, reducing the mean relative rotation error to 2.12° (from 11.35°).

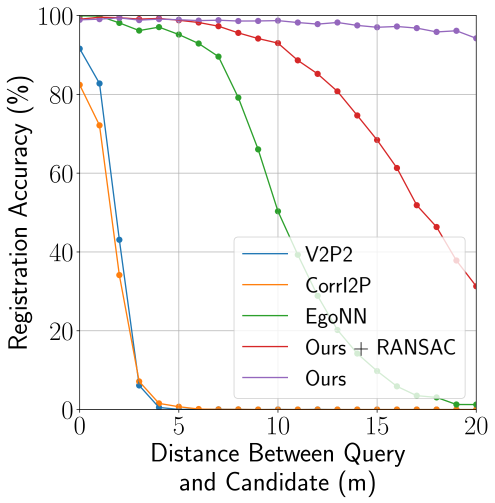
The authors back up the significantly better registration performance by showing that different to other base line methods, where performance degrades when the image and LiDAR pose don't align, SOLVR maintains good performance even when query and candidate views are far apart (Fig. 9).

2.3 SRL drone

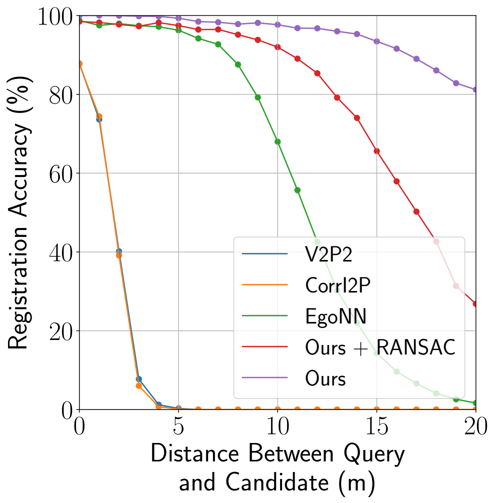
2.3.1 Multi-modal state estimation

Multi-modal sensors Smart Robotics Lab (SRL) drone built by SRL at TUM is a lightweight flying platform with only a visual-inertial sensor (Intel RealSense D455 [13]). The sensor outputs stereo images with $90\times65^\circ$ FoV and depth images with 2% depth accuracy within 4 m. Also, the visual-inertial sensor equips with a low-cost IMU, Bosch BMI055. We use depth from the sensor or deep neural networks [43, 42] for the presented estimator.

State estimation The state estimator is based on OKVIS2 [23] and its extension to depth networks [14]. The state is defined as in (1) and the optimization loss in the factor graph is same as (6). However, noisy depth measurements should be properly weighted for downstream tasks in contrast to precise LiDAR point clouds. The presented estimator is fully *uncertainty-aware* in the sense that occupancy integration and the visual-inertial factor graph are properly weighted by the estimated uncertainty



(a) KITTI-09



(b) KITTI-10

Figure 9: 6-DoF LiDAR-Visual Registration accuracy versus distance between query and candidate pairs on sequences KITTI-09 and 10.

of *depth fusion*. We optimally fuse depths from stereo and MVS networks based on their predicted uncertainties motivated by the complementary characteristics of small and large baselines from static and motion stereo.

It is pivotal to obtain a reliable dense depth as well as the associated uncertainty for our downstream tasks. Our key idea is to fuse static and motion stereo, which are complementary to each other — static stereo provides a reliable small baseline even when stationary, while motion stereo potentially brings a large baseline, depending on the camera motion. Our method does not depend on a specific network, but we found that Unimatch [43] and the MVS network [41] work well in real-world scenes. However, the previous networks only predict disparity or depth without any uncertainties. Therefore, we augment the base architecture with an uncertainty decoder and adopt the Laplacian loss function for (aleatoric) uncertainty learning. Specifically, our stereo network loss function is

$$\mathcal{L}_{\mathrm{st}}(\boldsymbol{\theta}) = \mathcal{L}_u(\boldsymbol{\theta}) + \mathcal{L}_{\nabla u_x}(\boldsymbol{\theta}) + \mathcal{L}_{\nabla u_y}(\boldsymbol{\theta}), \tag{10}$$

where $\boldsymbol{\theta}$ is the network weights, and u stands for the disparity. The disparity loss $\mathcal{L}_u$, modeled in the Laplacian distribution as in [16, 30], is defined as

$$\mathcal{L}_u(\boldsymbol{\theta}) = \sum_{i \in \mathcal{T}} \frac{\left|u_i - u_{\mathrm{gt}_i}\right|}{\sigma_{u_i}} + \log \sigma_{u_i}, \tag{11}$$

where $\mathcal{T}$ is a training set including pairs of stereo images and the ground-truth disparity. We additionally add the gradient loss $\mathcal{L}_{\nabla u}$ for sharper uncertainty output, which is analogously defined as $\mathcal{L}_u$. The gradient uncertainty along the horizontal and vertical directions is derived from the disparity uncertainty as

$$\begin{aligned} \sigma_{\nabla u_x} &= \sqrt{\sigma_u^2(x+1, y) + \sigma_u^2(x-1, y)}, \\ \sigma_{\nabla u_y} &= \sqrt{\sigma_u^2(x, y+1) + \sigma_u^2(x, y-1)}. \end{aligned} \tag{12}$$

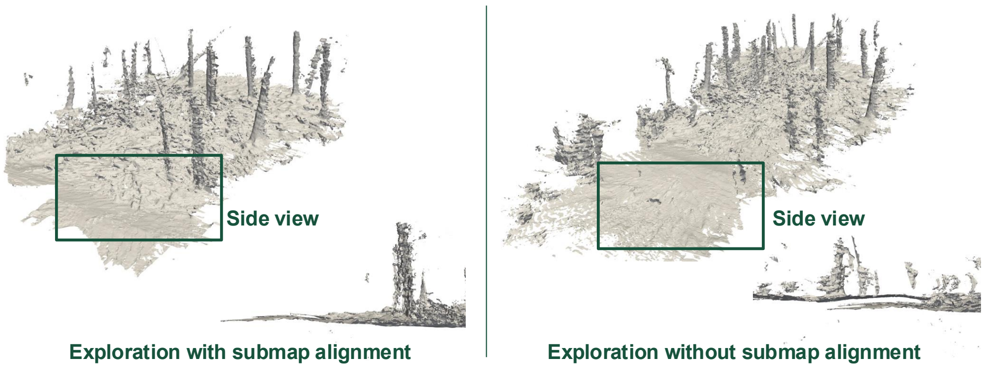

Side view
Exploration with submap alignment
Side view
Exploration without submap alignment

Figure 10: 3D mesh reconstructed onboard by SRL drone in the field trip, Stein am Rhein in July 2024. (Left) The first and the last submaps are well-aligned due to submap alignment factors in the factor graph optimization. (Right) The submaps are geometrically inconsistent without the submap alignment factor.

We propagate the uncertainty from the disparity to the depth with linearization,

$$\hat{d}_{\text{st}} = \frac{f_c b}{u}, \quad \sigma_{\text{st}} = \frac{f_c b}{u^2}\sigma_u, \tag{13}$$

where f_c is the rectified focal length, b is the stereo baseline.

Likewise, we modify the loss function of the MVS network [41]

$$\mathcal{L}_{\text{mvs}}(\phi) = \sum_{i \in \mathcal{T}} \frac{\left|\log d_i - \log d_{\text{gt}_i}\right|}{\sigma_{l_i}} + \log \sigma_{l_i}, \tag{14}$$

where the network learns log-depth uncertainty. We transform the log-depth to a depth with a linearized model,

$$\hat{d}_{\text{mvs}} = \exp(\log d_i), \quad \sigma_{\text{mvs}} = \hat{d}_{\text{mvs}}\sigma_l. \tag{15}$$

Given the pixel-wise estimates from the networks $(\hat{d}_{\text{st}}, \sigma_{\text{st}})$, $(\hat{d}_{\text{mvs}}, \sigma_{\text{mvs}})$ and with the assumption that two estimates are independent, we can optimally fuse two depth estimates [5],

$$\begin{aligned} \hat{d}_{\text{fuse}} &= \sigma^2_{\text{fuse}} \left(\sigma^{-2}_{\text{st}} \hat{d}_{\text{st}} + \sigma^{-2}_{\text{mvs}} \hat{d}_{\text{mvs}}\right), \\ \sigma^2_{\text{fuse}} &= \left(\sigma^{-2}_{\text{st}} + \sigma^{-2}_{\text{mvs}}\right)^{-1}. \end{aligned} \tag{16}$$

Fused depth naturally maintains the optimal depth based on uncertainty-aware fusion.

Experiments The presented estimator is quantitatively evaluated in the EuRoC dataset [3]. This dataset provides stereo images and IMU measurements recorded by a drone, the ground-truth trajectory and point clouds of the Vicon room in mm-level accuracy [3]. We use the stereo pair for our stereo network and left images for our MVS network. Table 4 summarizes ATE where all competitors also use a stereo and inertial configuration. Our method improves the baseline, OKVIS2 [23] by adding occupancy-to-point factors in causal evaluation. We obtained the same accuracy on average when compared to the baseline in non-causal evaluation. This indicates

Table 4: Absolute Trajectory Error [m] in the EuRoC dataset

	VINS-Fusion^1[32]	OKVIS2^2[23]	DVI-SLAM^3[29]	Ours	ORB-SLAM3^3[4]	OKVIS2^2[23]	Ours
	Causal				*Non-causal*		
MH01	0.166	0.034	0.042	0.034	0.036	0.021	**0.019**
MH02	0.152	0.029	0.046	0.031	0.033	0.020	**0.019**
MH03	0.125	0.039	0.081	0.037	0.035	0.032	**0.029**
MH04	0.280	0.065	0.072	0.072	**0.051**	0.059	0.071
MH05	0.284	0.087	0.069	0.081	0.082	**0.061**	0.066
V101	0.076	0.037	0.059	**0.035**	0.038	**0.035**	**0.035**
V102	0.069	0.031	0.034	0.028	**0.014**	**0.014**	**0.014**
V103	0.114	0.035	0.028	0.029	0.024	**0.020**	0.021
V201	0.066	0.039	0.040	0.033	0.032	**0.023**	0.024
V202	0.091	0.030	0.039	0.028	**0.014**	0.019	0.016
V203	0.096	0.041	0.055	0.041	0.024	0.027	**0.022**
Avg	0.138	0.043	0.051	0.041	0.035	**0.030**	**0.030**

^{1} Results taken from [4].
^{2} Results obtained ourselves.
^{3} Results taken from the paper.

that the depth fusion could not achieve 3 cm-level accuracy, which is already quite accurate. On the other hand, our approach outperforms state-of-the-art methods with nontrivial margins. Since our method tightly couples localization and volumetric mapping, we also evaluate the mapping accuracy in Table 5. For fair comparison to Simplemapping [41] in a monocular-inertial set up, we implement *Ours-Mono* where only left images are used without the stereo network. The proposed method achieves the highest accuracy and completeness, while *Ours-Mono* also outperforms the competitor.

The presented state estimator was employed during the field trip in Stein am Rhein (July 2024). Fig. 10 qualitatively shows reconstructed 3D mesh onboard in SRL drone during exploration. To be specific, depth measurements from RealSense were used in this experiment. This highlights the effectiveness of the submap alignment factor in the factor graph optimization. By adding the submap alignment factors in the optimization, a globally consistent map was estimated as in Fig. 10(left), while the two submaps were not well-aligned without the submap alignment factor in Fig. 10(right).

2.3.2 Detection and handling of sensor degradation and failure

The state estimator in SRL drone fuses multi-modal sensor measurements including IMU and camera measurements and shares the same architecture as in BLK2FLY where depth measurements replace LiDAR point clouds. For instance, the depth factor (5) is down-weighted, if the depth uncertainty is high or the log-odd gradient of the occupancy is low. Furthermore, the navigation stack is sensor-agnostic where the

Table 5: Mesh accuracy [m] without threshold and completeness [%] with 0.2m threshold in the EuRoC dataset

	Simplemapping[1][41]	Ours-Mono	Ours	Simplemapping[1][41]	Ours-Mono	Ours
	Accuracy			*Completeness*		
V101	0.176	<u>0.076</u>	**0.042**	40.58	<u>41.74</u>	**44.28**
V102	0.123	<u>0.073</u>	**0.041**	<u>55.07</u>	54.96	**58.38**
V103	0.161	<u>0.093</u>	**0.048**	47.74	<u>55.92</u>	**68.48**
V201	0.155	<u>0.090</u>	**0.072**	<u>41.93</u>	39.62	**43.64**
V202	0.156	<u>0.073</u>	**0.069**	<u>56.37</u>	54.79	**57.66**
V203	0.090	<u>0.089</u>	**0.072**	**62.58**	58.42	<u>59.96</u>
Avg	0.144	<u>0.082</u>	**0.057**	50.71	<u>50.91</u>	**55.40**

[1] Results obtained ourselves with the same metric.

software is compatible with depth as well as LiDAR sensors. For details, please refer to Sec. 2.2.2.

2.3.3 Place recognition and localization

A lightweight SRL drone, which is equipped with a stereo camera, would re-localize in the sparse LiDAR point clouds. Specifically, a submap built by depth measurements from a stereo camera is registered in the corresponding LiDAR scan by retrieving a corresponding place from the LiDAR scan database. This multi-modal relocalization works with BLK2FLY as explained in Sec. 2.2.3. For further details, please refer to Sec. 2.2.3.

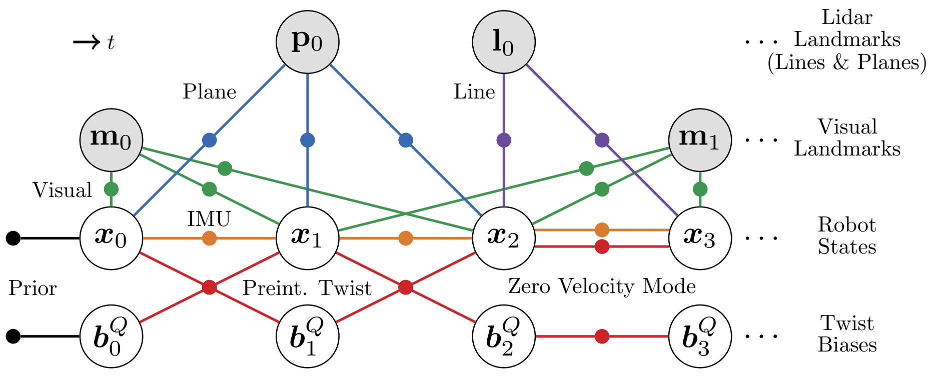


Figure 11: VILENS factor graph structure. The factors are: prior (black), visual (yellow), lidar planes (green), lidar lines (red), preintegrated IMU (orange), preintegrated velocity (from leg kinematics, blue), and lidar odometry (from ICP registration, magenta). State nodes are white, while landmarks are grey.

3 ANYmal – legged robot

In this section, we present VILENS (Visual Inertial Lidar Legged Navigation System), an odometry system for legged robots based on factor graphs. The key novelty is the tight fusion of four different sensor modalities to achieve reliable operation when the individual sensors would otherwise produce degenerate estimation. To minimize leg odometry drift, we extend the robot's state with a linear velocity bias term which is estimated online. This bias is observable because of the tight fusion of this preintegrated velocity factor with vision, lidar, and IMU factors.

3.1 Multi-modal state estimation

Multi-modal sensors Most legged robots are equipped with a high frequency (>250 Hz) proprioceptive state estimator for control and local mapping purposes. However, deformable terrain such as forest floors, leg flexibility, and foot slippage can degrade estimation performance up to the point where local terrain reconstruction is unusable and multi-step trajectories cannot be executed, even over short ranges. This problem is more evident when a robot is moving dynamically, and can be the limiting factor when crossing rough terrain.

In our work, we fuse all four sensor modalities (IMU, kinematics, lidar, camera) in a tightly coupled fashion. The quadruped robot has 12 active Degrees-of-Freedom (DoF) and is equipped with a stereo camera, an IMU, joint encoders and torque sensors. The goal is to estimate the history of the robot's base link pose and its velocity (linear and angular) over time.

The state estimation architecture is shown in Fig. 12. Four parallel threads execute the following operations: preintegration of the IMU factor, preintegration of the velocity factor, stereo feature tracking, and optimization. The factors are illustrated in Fig. 11.

This approach outputs 400 Hz velocity and pose estimates from the preintegration thread for use by the robot's control system, and a 30 Hz output from the factor-graph optimization thread for use by local mapping. When a new keyframe is processed, the preintegrated measurements and tracked landmarks are collected by the optimization

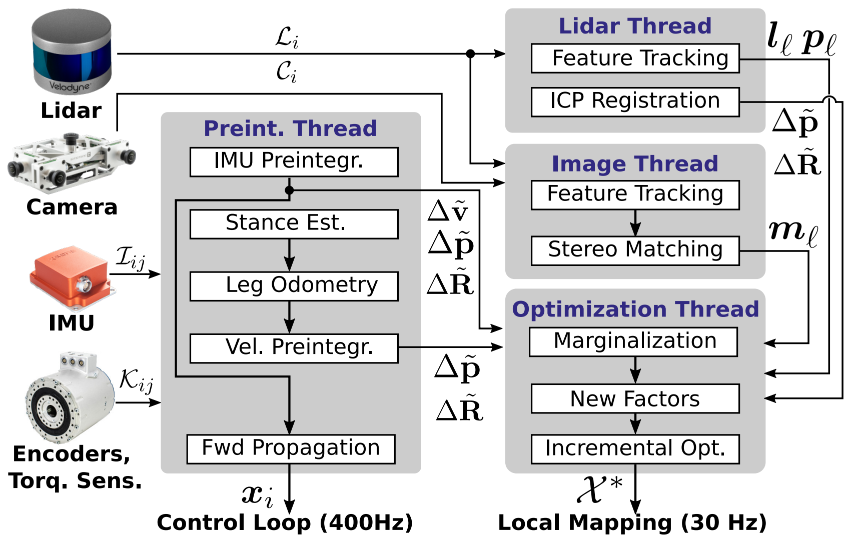


Figure 12: The VILENS architecture with preintegrated velocity bias estimation.

thread, while the other threads process the next set of measurements.

The factor graph optimization is implemented using the efficient incremental optimization solver iSAM2 [15], which is part of the GTSAM library [6]. We limit the number of states in the graph to 500 to keep the optimization time approximately constant.

Visual Feature Tracking

We detect features using the FAST corner detector, and track them between successive frames using the KLT feature tracker. Outliers are rejected using a RANSAC-based rejection method. Thanks to the parallel architecture and incremental optimization, all frames are used as keyframes, achieving 30 Hz nominal output.

Zero Velocity Update Factors

To limit drift and factor graph growth when the robot is stationary, we enforce zero velocity updates on the different sensor modalities (camera, IMU, and leg odometry). If two out of three modalities report no motion, a zero velocity constraint factor is added to the graph. The IMU and leg odometry threads report zero velocity when position (rotation) is less than 0.1 mm (0.5°) between two keyframes. The image thread reports zero velocity when less than 0.5 pixels displacement of all the features is detected over the same period.

Experiments We deploy our state estimation stack – VILENS – running onboard the ANYmal robot. The limited on-board computation is shared with other processes running on the robot, including control, terrain mapping, and sensor drivers. Constructing an accurate local terrain map around the robot is crucial for perceptive locomotion and path planning. Locomotion controllers plan footstep placements on these maps.

Fig. 13 shows the pipeline used for the terrain mapping with VILENS. The inputs to the terrain mapping module [7] are the VILENS state estimate, and the point clouds

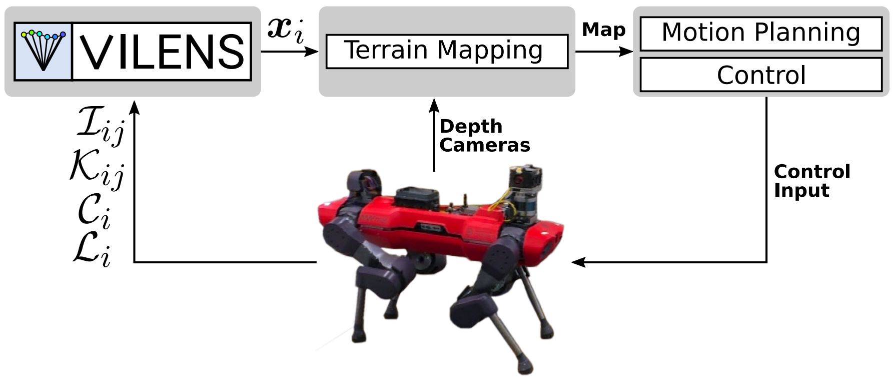


Figure 13: *Terrain Mapping Pipeline:* The VILENS state estimator produces a high frequency, low drift state estimate, for the terrain mapping module [7]. This local terrain map can then be used by other modules, such as perceptive motion planning and control [10], or local path planning for obstacle avoidance [26].

from several downward-facing depth cameras on the robot's body. These cameras are not the same as the ones used for state estimation and are not triggered at the same time. Therefore, accurate terrain mapping depends on a smooth, accurate, and high frequency state estimate to avoid interpolation errors.

From our experience, terrain mapping frequencies of $\geq 15\,\mathrm{Hz}$ are required for dynamic locomotion over rough terrain. This means that registration-based algorithms with a low frequency output (such as ICP or LOAM) are not suitable for this purpose.

3.2 Detection and handling of sensor degradation and failure

Handling Sensor Degradation

Multi-sensor SLAM systems, such as those used in the ANYmal legged robot, are susceptible to sensor degradation due to several factors. Below are some common sources of sensor degradation:

- *Unreliable/interruptive GNSS signal coverage under-canopy*: In dense forested environments, GNSS signals are often weak or entirely blocked due to thick vegetation. This leads to unreliable position estimates or total loss of signal.
- *Low-light conditions and sudden changes in exposure*: When the robot transitions from dense under-canopy regions to open areas, there can be a significant change in lighting conditions. Sudden exposure changes or low-light scenarios effect result in poor image quality and unreliable visual feature extraction.
- *Environmental disturbances:* Wind-induced movement of trees and branches can cause small but continuous changes in the surrounding environment. This can create difficulties in sensor data consistency, particularly for LiDAR and visual sensors that rely on environmental features for tracking and mapping.

Strategies to overcome sensor degradation

To ensure the robustness of state-estimation systems under these conditions, the following strategies are employed:

To mitigate the effects of sensor degradation and failures, the ANYmal's state-estimation system relies on multiple sensor modalities to provide redundancy. VILENS (described in Sec. 3.1) integrates multiple sensor types, including IMU, LiDAR, Camera, and leg odometry. The multi-sensor fusion approach provides robustness by ensuring that the failure or degradation of any single sensor does not lead to catastrophic localization or mapping failures.

The strategies to handle sensor degradation are based on a two-stage approach: outlier rejection and adaptive re-weighting of measurements.

- **Outlier Rejection**: The first stage in the pipeline is designed to filter out low-quality measurements before they are passed to the SLAM optimization algorithm. For each sensor type, specific criteria are used to identify degraded or outlier data:
  - Camera-based Visual Features: For visual sensors, the system monitors the quality of feature extraction by checking both the number and the distribution of visual features in each frame. If the number of features falls below a pre-set threshold, indicating issues like blurring, poor lighting, or low-texture scenes, the measurements are classified as outliers and are discarded from the SLAM process.
  - LiDAR Data: In environments where tree branches are swaying due to wind or where there are other moving elements in the scene, LiDAR measurements may become unreliable. To handle this, a threshold-based heuristic is used to reject inconsistent or highly variable point clouds.

  By rejecting low-quality data early, the system prevents poor sensor readings from corrupting the overall SLAM estimation process, thus improving the reliability of the final state estimate.
- **Adaptive re-weighting**: In the second stage, the system adjusts the weight of each sensor measurement in the SLAM optimization process, which is often modeled as a factor-graph. Each sensor's contribution to the state estimation is weighted based on the quality of its current measurements. This is done by modeling the covariance of each sensor's data.
  - Dynamic Covariance Scaling: For each sensor, the covariance associated with its measurements is dynamically adjusted based on the observed noise or error characteristics. For example, in the case of degraded GNSS signals under dense canopy, the system automatically increases the covariance, reducing its contribution to the final pose estimate, while giving more weight to the more reliable sensors like IMU or leg odometry.

This combination of outlier rejection and adaptive re-weighting ensures that the SLAM system can continue to operate effectively even in challenging environments where one or more sensors may be temporarily degraded or completely fail. By fusing information from diverse modalities, such as IMU for short-term stability, LiDAR and visual features for large-scale localization, and leg odometry for reliable proprioception, the ANYmal robot is able to maintain accurate localization and mapping despite sensor degradations

3.3 Place recognition and localization

In this section, we describe the place recognition system and its integration with the localization stack on the ANYmal robot. The system was developed with focus

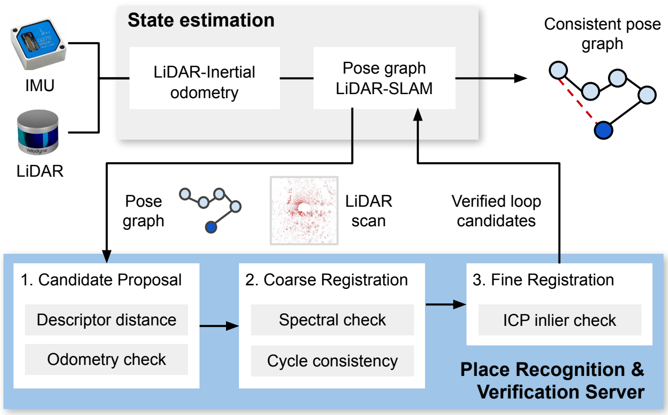


Figure 14: Place recognition pipeline. VILENS provides a continuous odometry estimate at 10 Hz. Pose graph SLAM is used to optimize poses after successful loop closure verification. The place recognition module consists of three steps: loop candidate proposal, coarse registration, and fine registration. We verify the loop candidates at the global descriptor-level, local feature consistency -level, and finally at the fine registration level. A loop candidate is integrated in the pose graph only if it passes these three stages.

on being generally useful for multiple platforms – based around LiDAR data. We deployed the same place recognition system for the backpack-based mapping system in addition to the ANYmal platform as well.

The overall system infrastructure is shown in Fig. 14. For state estimation, we use a LiDAR-inertial odometry system—VILENS [40]—, in conjunction with a pose graph SLAM framework, described previously in Sec. 3.1. We implemented a *place recognition & verification server*, which not only provides a common interface for the different LiDAR-based place recognition models but also multi-stage verification procedures for its use in the different settings.

In the following sections, we introduce the technical details of the place recognition server, and present its integration to state-estimation stack.

Step 1: Loop candidate proposals

Initial loop closure candidates are obtained by comparing global descriptors extracted from the pose graph scans as well as the query scan. We evaluates four state-of-the-art methods for descriptor extraction: the learning-based Logg3dNet [37] and EgoNN [21], as well as the handcrafted ScanContext [18] and STD [45].

Given the reference pose graph and the query scan, we compute a database of descriptors using all the scans in the pose graph, given by the matrix $\mathbf{D} \in \mathbb{R}^{N\times M}$, where N is the number of poses in the pose graph and M the descriptor dimension. Additionally, we compute the descriptor for the query scan, denoted by $\mathbf{d}_q \in \mathbb{R}^{M\times 1}$.

To obtain candidates, we compute the pairwise distances of the scan to the database using the cosine similarity:

$$\mathbf{S} = \mathbf{D} \cdot \mathbf{d}_q \in \mathbb{R}^{N\times 1} \tag{17}$$

The vector of descriptor distances $\mathbf{S}$ is sorted by increasing distance, and only the top-k candidates are selecting using a distance threshold τ_s, which is set by the F_1-max score from testing data.

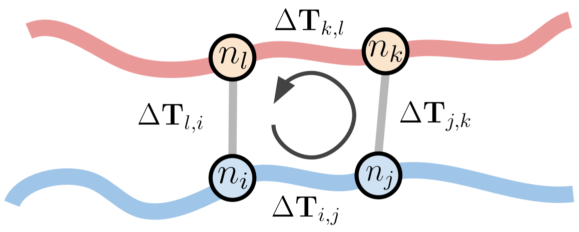


Figure 15: The cycle consistency check uses the relative transformation estimates and loop candidates between four nodes n_i, n_j, n_k, n_l to verify the validity of a loop.

Step 2: Coarse Registration

Next, we estimate the relative 6DoF transformation that expresses the pose associated to the query scan w.r.t each candidate node, which we denote $\Delta\mathbf{T}$. For the handcrafted methods (ScanContext and STD), the relative transformation is directly an output of descriptor computation. For the learning-based approaches, we use the point-wise feature vectors outputted in the forward pass of Logg3dNet and EgoNN for feature matching, which is used in a RANSAC-based pose estimation scheme [8] to estimate the relative transformation.

We additionally verify the inlier matches using the *Spectral Geometric Verification* [38] method, which provides an additional measure of the quality of the feature matches.

Lastly, we carry out a *cycle consistency* verification (Fig. 15), which checks whether the relative transformations between pairs of nodes are mutually consistent with one another. Given four pose graph nodes n_i, n_j, n_k, n_l, we test how close the following equivalence holds:

$$\Delta\mathbf{T}_{i,j}\,\Delta\mathbf{T}_{j,k}\,\Delta\mathbf{T}_{k,l}\,\Delta\mathbf{T}_{l,i} \approx \mathbf{I}_{4\times 4} \tag{18}$$

If this difference is more than a threshold of 10 cm or 1° we reject the candidate. This is shown in Fig. 15.

Step 3: Fine Registration

Finally, we employ the Iterative Closest Point (ICP) algorithm [1] for fine registration of the proposed candidates. We use the *libpointmatcher* implementation [31], which also provides information on the quality of the registration, such as the proportion inlier points and the residual error of each point to access the alignment.

3.3.1 Integration with the State-Estimation and SLAM System

The first task we consider is LiDAR-based online SLAM. Our implementation defines it as an incremental pose graph estimation problem (see Fig. 16,(a)). Consider consecutive loop closures at nodes n_i, n_{i+1} and n_j, n_{j+1}. Edges are provided by relative estimates from our LiDAR-inertial odometry system (odometry factors, denoted by $\Delta\mathbf{T}_{i,i+1}, \Delta\mathbf{T}_{j,j+1}$), and verified loop closure candidates from our place recognition server (loop closure factors, $\Delta\mathbf{T}_{i+1,j}, \Delta\mathbf{T}_{i,j+1}$).

For the cycle consistency verification described in Sec. 3.3, we consider the relative transformation change between consecutive loop closure candidates w.r.t the pose graph poses and the odometry change (Fig. 15 i, j, k, l, replaced by i, $i+1$, j, $j+1$).

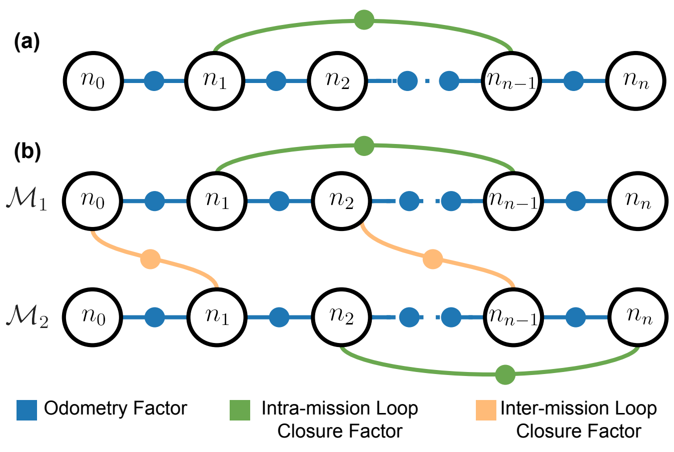


Figure 16: Pose graph formulation used for (a) online, and (b) offline multi-mission SLAM optimization. Each node n_i has a 6DOF pose $\mathbf{x}_i$, which correspond to the main variables estimated on each case.

Again, a cycle consistency needs to be satisfied:

$$\Delta\mathbf{T}_{i,i+1}\,\Delta\mathbf{T}_{i+1,j}\,\Delta\mathbf{T}_{j,j+1}\,\Delta\mathbf{T}_{i,j+1}^{-1} \approx \mathbf{I}_{4\times 4} \tag{19}$$

3.3.2 Evaluating Place Recognition Descriptors

Our evaluation included four distinct test sites featuring varying forest compositions captured by two different LiDARs. We used a Hesai XT32 LiDAR—50 m effective range and 30° narrow field of view and a Hesai QT64—30 m range, 100° wide field of view.

- Evo (Finland), Hesai XT-32, characterized by tall, a mix of broad-leaf and coniferous trees.
- Stein am Rhein (Switzerland), Hesai XT-32, thinned coniferous trees, open canopy, and flat ground.
- Wytham Woods (UK), Hesai QT-64, a very dense forest with cluttered trees with mixed species and ground vegetation, as well as uneven terrain (valleys and hills).
- Forest of Dean (UK), Hesai QT-64, less dense broad-leaf, oak trees and flat grounds.

In these experiments, we evaluated the descriptors of four different place recognition models (Logg3dNet, EgoNN, ScanContext, STD) focusing on their ability to accurately capture loop-candidates in forest environments. Logg3dNet and EgoNN models are learning-based methods and were pre-trained on the Wild-Places dataset. Our experiment measured the precision-recall curves to assess how well each model detected correct loop candidates within a reasonable distance threshold (here set to 10 m). We measured this at various descriptor distance thresholds τ_s in four different forests.

From our obtained precision-recall curves Fig. 17, it is evident that Logg3dNet consistently outperforms the other models across the four different forests. Particularly, on the Evo and Stein am Rhein datasets, Logg3dNet showed the best performance both

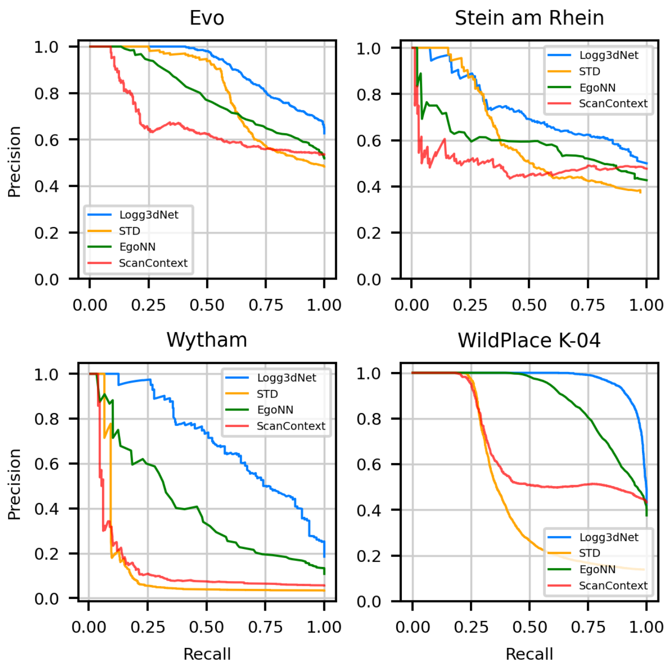


Figure 17: Precision-recall curves obtained for Logg3dNet, STD, EgoNN and Scan Context in our four dense forest datasets. Only the top-1 candidate within 10 m of the ground truth position is regarded as a true positive candidate.

in terms of precision and recall, without experiencing any drastic drops in precision. In contrast, ScanContext demonstrated a significant decrease in precision, attributed to its dependency on the vertical field of view of the input scan. In more challenging scenarios, such as Wytham Woods, handcrafted models showed a notable decline in performance. However, Logg3dNet remained at the top, successfully retrieving a substantial portion of correct loop-candidates, achieving a 70% precision at a 50% recall rate.

3.3.3 Evaluating the Online Place Recognition Pipeline

In this experiment, we investigated the online place recognition capability of our system, wherein loop closures from the place recognition module are integrated into the SLAM system. The database **D**, is incrementally built as the robot moves through the environment. When matching, we exclude the most recent 30 seconds of data to prevent loop closures with immediately recent measurements.

Fig. 18 presents an illustrative example of online SLAM performance on the Evo dataset, depicting the sets of loop candidates after each verification step. Initially, many loop closure candidates are proposed (shown in blue) under a descriptor matching threshold τ_s of F_1-max score. Loop closures beyond a conservative estimate of 20 m are rejected using the odometry information. After this, a subset of loop closure candidates are identified using RANSAC matching (highlighted in orange), and finally, a refined set of loop closures that pass the consistency and ICP steps are integrated into the SLAM framework. Final loop closures (shown in red) are one of ICP verified loop candidates by checking pose graph density to avoid over-constraining the pose graph.

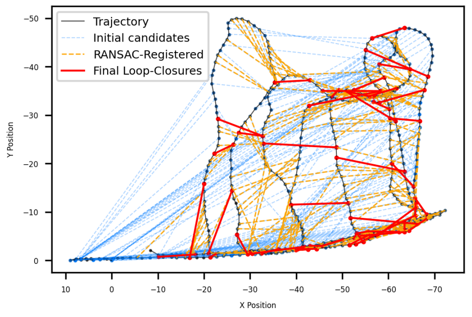


Figure 18: Online place recognition on a sequence of the Evo dataset. Bold red lines show the loop closures integrated into the SLAM system, successfully identified up to distance of 17 m within dense forest areas.

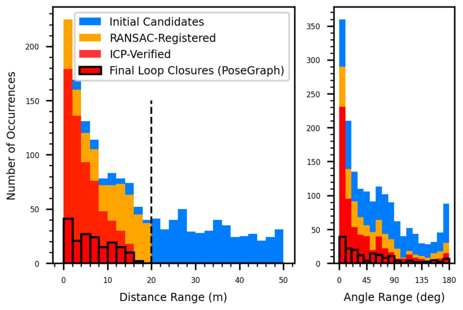


Figure 19: Loop closures distribution by distance and angle at various stages of the pipeline on Evo dataset. Initial candidates based on descriptor distance are shown in blue. Candidates beyond 20 m are rejected using odometry information. Candidates within 20 m undergo RANSAC pre-registration with additional verification steps of SGV[38] and pairwise checks (Yellow). Then these candidates are refined using ICP fine-registration (Red), and final loop closures in the pose graph after checking constraints density in pose graph (red with black outlined).

Next, we conducted a comprehensive analysis of loop closure statistics based on distance and viewpoint angles, shown Fig. 19. Our findings show that the system can successfully identify loop closure pairs across considerable baseline distances (10 m to 20 m). We observed that despite the large baseline, a significant portion of initial candidates can be registered using RANSAC-based matching, indicating that the correspondences are accurate. However, the proportion of candidates verified by ICP decreases as the distance between scans increases. Specifically, when scans are 10 m apart, ~60 % of RANSAC-registered candidates are successfully verified by ICP, and when 15 m apart, only ~40 % remains verified. This decrease is due to the diminishing overlap ratio between corresponding scans with increasing distance, making convergence of ICP challenging.

Similarly, in terms of viewpoint orientation difference, we observed that a large proportion of the loop candidates up to 90° difference are verified both at the RANSAC-registration and ICP-based checks. However, we observed a degradation of performance over 90°, which can be attributed to the occlusions in the scans present at large orientation differences. Nonetheless, despite this degradation, the number of final loop closures integrated into the SLAM system proved to be sufficient for correcting the drift.

4 SAHA – harvester

4.1 Multi-modal state estimation

On the autonomous harvester, SAHA, we implemented the Graph-based Multi-Sensor Fusion (Graph-MSF) method. The method is designed for robust and consistent localization of large-scale autonomous construction, farming, and forestry robots, specifically targeting real-world operational environments. In such large-scale outdoor operations, GNSS usually provides a rough reference for global localization, although its quality might suffer from noise and dropouts caused by the canopy in a forest environment. Therefore, a major challenge in the large-scale autonomous operation of autonomous forest machines is maintaining precise localization across different terrains and under sensor failures or dropouts. The Graph-MSF method addresses these issues by combining filtering and smoothing techniques within a graph-based framework, allowing for accurate pose estimation even when certain sensors (e.g., GNSS) become unreliable. Graph-MSF has been verified on walking excavators [27], where it showed accurate and consistent localization. In DigiForest, we implemented it on SAHA, which has similar sensor modalities, such as IMU, LiDAR, and GNSS. Instead of performing place recognition with onboard sensors, we mainly rely on GNSS for globally consistent localization when SAHA drives on forest roads. Once SAHA enters the dense forests and GNSS becomes unreliable, local pose estimations with LiDAR odometry can be used. Through extensive experiments in forest environments, the algorithm's great robustness and adaptability to different environments are demonstrated.

Graph-MSF employs a dual-graph structure with the following key elements:

- Graph-based prediction-update loop: Integrates both filtering and optimization-based methods.
- Asynchronous sensor fusion: Handles different update rates and sensor dropouts using a flexible framework.
- Dual-graph design: Allows for smooth transitions between global and local pose estimates during sensor outages (e.g., GNSS dropouts).

This method focuses on maximizing robustness during sensor failures while minimizing computational overhead, providing real-time localization suitable for control tasks like chassis balancing and navigation.

4.1.1 Mathematical Formulation

The localization problem is framed as an optimization task over sensor data, which is processed in two steps:

1. **Prediction**: Incorporating incoming IMU measurements to propagate the current state estimates.
2. **Update**: Triggering optimization using the factor graph when additional sensor data (GNSS or LiDAR) arrives.

State Definition The robot state at time t_i, denoted as x_i, is composed of:

$$x_i = [I_{x_i}, T_i^{WO}, B_{x_i}]$$

Where:

- I_{x_i}: IMU state with respect to the world frame.
- T_i^{WO}: Transformation between the local odometry frame O and the world frame W.
- B_{x_i}: Base frame state with respect to O.

The method uses a *factor graph* approach, where IMU measurements are incorporated as high-frequency inputs, providing low-latency state estimates.

Measurement Model

- **IMU factors**: Formed by integrating linear acceleration and angular velocity measurements.
$$r_{II_i} = [r_{\Delta R_i}, r_{\Delta v_i}, r_{\Delta p_i}]$$
- **GNSS measurements**: When GNSS is available, it provides global positioning information integrated into the factor graph as *GNSS factors*.
$$r_{GG_i} = p_{WI}^{W} - \tilde{p}_{WI}^{W}$$
- **LiDAR measurements**: LiDAR is integrated as either an *odometry factor* (for relative pose) or a *pseudo-global factor* when GNSS data is unavailable.

4.1.2 Dual-Graph Design

The **dual-graph structure** is central to Graph-MSF's ability to handle sensor failures:

- **Main Graph**: Used when GNSS data is available. LiDAR measurements are incorporated as odometry factors, and GNSS ensures global pose accuracy.
- **Fallback Graph**: Activated when GNSS fails. The LiDAR measurements are treated as pseudo-global factors to maintain local consistency and avoid drift in state estimation.

During GNSS outages, the fallback graph continues to estimate the pose using IMU and LiDAR, preserving smooth localization. When GNSS becomes available again, the main graph resumes, and any accumulated drift is corrected.

4.1.3 Prediction-Update Loop

Graph-MSF introduces a **prediction-update loop** that ensures:

- **Fast updates**: IMU measurements are processed without triggering a full graph optimization, thus maintaining real-time operation.
- **Optimization updates**: The graph optimization is triggered only when additional (delayed) sensor measurements such as GNSS or LiDAR are available.

This hybrid approach balances the need for fast, low-latency state estimation with the global consistency provided by graph-based smoothing.

4.2 Detection and handling of sensor degradation and failure

A critical feature of the Graph-MSF method is its ability to maintain accurate and consistent localization even in the presence of sensor degradation and failure, particularly with GNSS (Global Navigation Satellite System) signals. Forest environments are often subject to conditions where sensors can experience temporary degradation or failure due to dense tree cover. Graph-MSF is designed to effectively manage these challenges by implementing a robust multi-sensor fusion approach.

Dual-Graph Architecture for Robustness The key to addressing sensor degradation in Graph-MSF lies in its **dual-graph architecture**, which enables smooth transitions between global and local localization estimates depending on the availability and reliability of sensor inputs. The method introduces two distinct factor graphs:

- **Main Graph**: This graph processes all sensor inputs when GNSS data is available and reliable. It uses GNSS for global positioning and incorporates LiDAR measurements as relative odometry factors, ensuring accurate global localization.
- **Fallback Graph**: When GNSS measurements become unreliable or unavailable due to sensor degradation, the system switches to the fallback graph. In this mode, the LiDAR data is treated as a pseudo-global factor, allowing the system to continue generating consistent local pose estimates. This fallback graph is crucial for maintaining continuity in localization during GNSS dropouts.

During periods of GNSS failure, the fallback graph preserves localization consistency by using the onboard LiDAR and IMU sensors. Once GNSS signals are restored, the system seamlessly transitions back to the main graph, correcting any accumulated drift that may have occurred during the GNSS outage. The smooth recovery is ensured by estimating the relative transformation between the world and odometry frames, thus avoiding any sudden jumps in the robot's pose estimation that could disrupt control tasks.

Outlier Rejection and Sensor Data Reliability Graph-MSF includes additional mechanisms to detect and handle sensor data degradation. GNSS outliers, which can result from poor satellite coverage or interference of dense tree canopy, are identified using the covariance estimates provided by the GNSS system. Only transformations that meet strict thresholds for velocity and measurement consistency are accepted.

The system employs the following criteria to reject outlier data:

- **Covariance thresholding**: Modern RTK GNSS sensors provide covariance estimates that are directly used to assess the reliability of GNSS data.
- **Measurement consistency checks**: Delta transformations are monitored over time, and sudden, unexpected shifts are flagged as outliers if they exceed a predefined velocity threshold.

By employing these criteria, the system is able to maintain high-quality localization even in degraded sensor conditions. The fallback graph provides a buffer against sudden loss of global positioning, using LiDAR and IMU measurements to ensure continuity.

Recovery from GNSS Loss A major challenge in autonomous operations is ensuring smooth localization during the transition between sensor loss and recovery. When GNSS signals are restored after a dropout, there can be significant discrepancies between the locally estimated pose and the true global pose due to drift. Graph-MSF handles this through a smooth update mechanism that re-aligns the local odometry frame with the global world frame upon GNSS recovery.

The transformation from the odometry frame back to the world frame is computed as:

$$T^{WO} = T^{WI} \cdot (T^{OI})^{-1}$$

Where:

- T^{WI}: The global estimate from the world to the IMU frame.
- T^{OI}: The local odometry estimate from the odometry frame to the IMU frame.

This adjustment ensures that the fallback graph is reset to the state of the main graph once GNSS becomes reliable again, eliminating potential discontinuities in the robot's estimated position.

Consistent Localization during Sensor Dropouts One of the most important benefits of the Graph-MSF approach is its ability to maintain consistent localization even during extended periods of sensor dropout. By relying on the fallback graph, which uses LiDAR as a pseudo-global factor, the system ensures that the robot remains localized in its environment, avoiding the accumulation of drift often seen in purely odometry-based methods.

In summary, the Graph-MSF method provides a comprehensive solution for addressing sensor degradation and failure through its dual-graph architecture, robust outlier rejection mechanisms, and smooth transition handling between local and global localization frames. This enables reliable and consistent performance in the unpredictable and dynamic environments typical of large-scale construction tasks.

5 Summary

In conclusion, this deliverable has presented platform-specific state estimators in the aspects of multi-modality, sensor degradation/failure detection, and place recognition and localization. Specifically, aerial robots consist of NTNU RMF, Leica BLK2FLY, and TUM SRL drone. The state estimator in RMF is based on CompSLAM [17] that couples LiDAR odometry and VIO in a loosely coupled manner to ensure continuity of observations. In BLK2FLY and SRL drone, the state estimator is based on OKVIS2 [23] where the state estimator in BLK2FLY tightly couples multi-modal sensors including LiDAR, and SRL drone, which is a light-weight platform with only visual-inertial sensor, tightly fuses visual and inertial observations. For ground robots, the state estimator in ANYmal is based on VILENS [40] that tightly fuses four different sensor modalities including IMU, camera, LiDAR, and joint encoders. The state estimator in SAHA robot is based on Graph-MSF that combines filtering and smoothing in a graph-based framework with multi-modality including IMU, LiDAR, and a GNSS receiver. Future work will focus on demonstrating robot autonomy based on the state estimators where we have shown their robustness and accuracy throughout this document.

References

[1] Paul J. Besl and Neil D. McKay. “A Method for Registration of 3-D Shapes”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791. URL: https://doi.org/10.1109/34.121791.

[2] Simon Boche, Sebastián Barbas Laina, and Stefan Leutenegger. “Tightly-Coupled LiDAR-Visual-Inertial SLAM and Large-Scale Volumetric Occupancy Mapping”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2024).

[3] Michael Burri et al. “The EuRoC micro aerial vehicle datasets”. In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1157–1163.

[4] Carlos Campos et al. “ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[5] Neal A Carlson. “Federated square root filter for decentralized parallel processors”. In: *IEEE Transactions on Aerospace and Electronic Systems* 26.3 (1990), pp. 517–525.

[6] Frank Dellaert and Michael Kaess. *Factor Graphs for Robot Perception*. Vol. 6. 1-2. 2017. ISBN: 9781680833263. DOI: 10.1561/2300000043. arXiv: arXiv:1408.0952v2.

[7] Péter Fankhauser, Michael Bloesch, and Marco Hutter. “Probabilistic terrain mapping for mobile robots with uncertain localization”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3019–3026.

[8] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (1981), 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[9] Christian Forster et al. “On-manifold preintegration for real-time visual–inertial odometry”. In: *IEEE Transactions on Robotics* 33.1 (2016), pp. 1–21.

[10] J. Frey et al. “Fast Traversability Estimation for Wild Visual Navigation”. In: *Robotics: Science and Systems*. Daegu, Republic of Korea, 2023. DOI: 10.15607/RSS.2023.XIX.054.

[11] Nils Funk et al. “Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3553–3560.

[12] Akshay Hinduja, Bing-Jui Ho, and Michael Kaess. “Degeneracy-Aware Factors with Applications to Underwater SLAM”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1293–1299. DOI: 10.1109/IROS40897.2019.8968577.

[13] *Intel RealSense Depth Camera D455*. Sept. 2024. URL: https://www.intelrealsense.com/depth-camera-d455.

[14] Jaehyung Jung et al. “Uncertainty-Aware Visual-Inertial SLAM with Volumetric Occupancy Mapping”. In: 2024. URL: https://www.arxiv.org/abs/2409.12051.

[15] Michael Kaess et al. “ISAM2: Incremental smoothing and mapping using the Bayes tree”. In: *International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[16] Alex Kendall and Yarin Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems* 30 (2017).

[17] Shehryar Khattak et al. “Complementary Multi–Modal Sensor Fusion for Resilient Robot Pose Estimation in Subterranean Environments”. In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2020, pp. 1024–1029. DOI: 10.1109/ICUAS48674.2020.9213865.

[18] Giseop Kim and Ayoung Kim. “Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2018.

[19] Joshua Knights et al. “SOLVR: Submap Oriented LiDAR-Visual Re-Localisation”. In: 2024. URL: https://www.arxiv.org/abs/2409.10247.

[20] Jacek Komorowski, Monika Wysoczanska, and Tomasz Trzcinski. “Egonn: Egocentric neural network for point cloud based 6dof relocalization at the city scale”. In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 722–729.

[21] Jacek Komorowski, Monika Wysoczanska, and Tomasz Trzcinski. “EgoNN: Egocentric Neural Network for Point Cloud Based 6DoF Relocalization at the City Scale”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 722–729. DOI: 10.1109/LRA.2021.3133593.

[22] Junwoon Lee et al. “Switch-SLAM: Switching-Based LiDAR-Inertial-Visual SLAM for Degenerate Environments”. In: *IEEE Robotics and Automation Letters* 9.8 (2024), pp. 7270–7277. DOI: 10.1109/LRA.2024.3421792.

[23] Stefan Leutenegger. “OKVIS2: Realtime Scalable Visual-Inertial SLAM with Loop Closure”. In: *arXiv preprint arXiv:2202.09199* (2022).

[24] S Lynen et al. “A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation”. In: *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 2013.

[25] F Landis Markley et al. “Averaging quaternions”. In: *Journal of Guidance, Control, and Dynamics* 30.4 (2007), pp. 1193–1197.

[26] M. Mattamala, N. Chebrolu, and M. Fallon. “An Efficient Locally Reactive Controller for Safe Navigation in Visual Teach and Repeat Missions”. In: *IEEE Robotics and Automation Letters*. 2022.

[27] Julian Nubert, Shehryar Khattak, and Marco Hutter. “Graph-based Multi-sensor Fusion for Consistent Localization of Autonomous Construction Robots”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 10048–10054. DOI: 10.1109/ICRA46639.2022.9812386.

[28] *Ouster OS0-64*. Sept. 2024. URL: https://ouster.com/products/hardware/os0-lidar-sensor.

[29] Xiongfeng Peng et al. “DVI-SLAM: A dual visual inertial SLAM network”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2024), pp. 12020–12026.

[30] Matteo Poggi et al. “On the uncertainty of self-supervised monocular depth estimation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 3227–3237.

[31] François Pomerleau et al. “Comparing ICP Variants on Real-World Data Sets”. In: *Autonomous Robots* 34.3 (2013), pp. 133–148.

[32] Tong Qin et al. “A general optimization-based framework for global pose estimation with multiple sensors”. In: *arXiv preprint arXiv:1901.03642* (2019).

[33] Milad Ramezani et al. “Wildcat: Online continuous-time 3d lidar-inertial slam”. In: *arXiv preprint arXiv:2205.12595* (2022).

[34] Siyu Ren et al. "CorrI2P: Deep image-to-point cloud registration via dense correspondence". In: *IEEE Transactions on Circuits and Systems for Video Technology* 33.3 (2022), pp. 1198–1208.

[35] Sai Shubodh et al. "Lip-loc: Lidar image pretraining for cross-modal localization". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 948–957.

[36] Emanuele Vespa et al. "Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping". In: *IEEE Robotics and Automation Letters* 3.2 (Apr. 2018), pp. 1144–1151. ISSN: 2377-3766.

[37] Kavisha Vidanapathirana et al. "LoGG3D-Net: Locally guided global descriptor learning for 3D place recognition". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2022.

[38] Kavisha Vidanapathirana et al. "Spectral Geometric Verification: Re-Ranking Point Cloud Retrieval for Metric Localization". In: *IEEE Robotics and Automation Letters* 8.5 (2023), pp. 2494–2501.

[39] David Wisth, Marco Camurri, and Maurice Fallon. "VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots". In: *IEEE Transactions on Robotics* 39.1 (2022), pp. 309–326.

[40] David Wisth, Marco Camurri, and Maurice Fallon. "VILENS: Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots". In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 309–326. DOI: 10.1109/TRO.2022.3193788.

[41] Yingye Xin et al. "SimpleMapping: Real-time visual-inertial dense mapping with deep multi-view stereo". In: *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2023, pp. 273–282.

[42] Gangwei Xu et al. "Accurate and efficient stereo matching via attention concatenation volume". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.4 (2023), pp. 2461–2474.

[43] Haofei Xu et al. "Unifying flow, stereo and depth estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[44] Huaiyuan Xu et al. "C2L-PR: Cross-modal Camera-to-LiDAR Place Recognition via Modality Alignment and Orientation Voting". In: *IEEE Transactions on Intelligent Vehicles* (2024).

[45] Chongjian Yuan et al. "STD: Stable Triangle Descriptor for 3D place recognition". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[46] Ji Zhang, Michael Kaess, and Sanjiv Singh. "On degeneracy of optimization-based state estimation problems". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 809–816. DOI: 10.1109/ICRA.2016.7487211.

[47] Ji Zhang, Sanjiv Singh, et al. "LOAM: Lidar odometry and mapping in real-time." In: *Robotics: Science and systems*. Vol. 2. 9. 2014, pp. 1–9.

[48] Lintong Zhang et al. "Hilti-oxford dataset: A millimeter-accurate benchmark for simultaneous localization and mapping". In: *IEEE Robotics and Automation Letters* 8.1 (2022), pp. 408–415.

[49] Chunran Zheng et al. *FAST-LIVO: Fast and Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry*. 2022. arXiv: 2203.00893 [cs.RO].

[50] Junsheng Zhou et al. "Differentiable registration of images and lidar point clouds with voxelpoint-to-pixel matching". In: *Advances in Neural Information Processing Systems* 36 (2024).