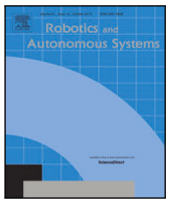
Robotics and Autonomous Systems 184 (2025) 104852

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Delta- and Kalman-filter designs for multi-sensor pose estimation on spherical mobile mapping systems☆

Fabian Arzberger [a,*], Tim Schubert [a], Fabian Wiecha [a], Jasper Zevering [a], Julian Rothe [b], Dorit Borrmann [c], Sergio Montenegro [b], Andreas Nüchter [a]

[a] Chair of Computer Science XVII – Robotics, Am Hubland, 97074, Würzburg, Germany
[b] Chair of Computer Science VIII – Aerospace Information Technology, Am Hubland, 97074, Würzburg, Germany
[c] THWS Robotics, Münzstraße 12, 97070, Würzburg, Germany

ARTICLE INFO

Keywords:
Spherical robots
Pose estimation
Sensor fusion
Kalman filter
Delta filter
Mobile mapping
LiDAR

ABSTRACT

Spherical mobile mapping systems are not thoroughly studied in terms of inertial pose estimation filtering. The underlying inherent rolling motion introduces high angular velocities and aggressive system dynamics around all principal axes. This motion profile also needs different modeling compared to state-of-the-art competitors, which heavily focus on more rotationally-restricted systems such as UAV, handheld, or cars. In this work we compare our previously proposed "Delta-filter", which was heavily motivated by the sensors inability to provide covariance estimations, with a Kalman-filter design using a covariance model. Both filters fuse two 6-DoF pose estimators with a motion model in real-time, however the designs are theoretically suitable for an arbitrary number of estimators. We evaluate the trajectories against ground truth pose measurement from an OptiTrack™ motion capturing system. Furthermore, as our spherical systems are equipped with laser-scanners, we evaluate the resulting point clouds against ground truth maps available from a Riegl VZ400 terrestrial laser-scanner (TLS). Our source code and datasets can be found on github (Arzberger, 2023).

1. Introduction

Spherical mobile mapping systems are just coming of age, as current research in the robotics community shows: The majority of research dealing with spherical systems is about locomotion mechanisms, e.g. [2–7]. Using spherical robots for mobile mapping (see Fig. 1) is a rather novel field. To the best of our knowledge, Borrmann et al. [8] first used a 2D laser-scanner mounted on a unicycle's wheel axis, to generate maps via offline—simultaneous localization and mapping (SLAM). In a follow-up study from our own lab [9] we used the same laser-scanner inside a spherical robot with a protective outer plastic shell. The robot is capable of self-initiated motion via flywheels utilizing an IBCOAM (impulse by conservation of angular momentum) approach. The idea of using spherical robots for mapping was explored in more depth by the European Space Agency (ESA) in 2021 during a concurrent design facility (CDF) study. This CDF study considers the general concept of a spherical robot for environment mapping and exploring lunar caves, but also terrestrial vents, to be feasible [10,11]. Another recent study [12] comes to a similar conclusion and substantiate the potential of using internal sensor such as LiDAR or cameras inside the spherical robot. Advantages of using spherical robots are a shell that protects internal sensors and a versatile locomotion mechanism that inherently results in sensor rotation leading to optimal coverage of the environment in all directions. In contrast to other protective shells, the spherical shape enables access to scenarios where other robotic systems could not operate, e.g., steep tunnels, underground mines, narrow funnels, or other hazardous environments. Furthermore, state-of-the-art 360° horizontal field of view (FoV) LiDAR sensors such as the 'Livox Mid-360', or 'Hesai Pandar-XT32' have only a limited vertical FoV. Thus, rolling such a LiDAR sensor leads to more coverage of the environment, especially considering the ceiling and ground. Due to the inherent rolling mechanism for locomotion, additional actuators, which potentially get damaged, are not needed to rotate the LiDAR sensor. However, during SLAM, large and aggressive rotations are the least favorable motions that a mobile mapping system could experience. This is because for any falsely estimated translation, the point-to-point errors in the resulting environment grow linearly, whereas for rotation these

☆ We acknowledge funding from the European Space Agency (ESA), Germany Contract No. 4000130925/20/NL/GLC for the study "DAEDALUS – Descent And Exploration in Deep Autonomy of Lava Underground Structures" within the Open Space Innovation Platform (OSIP) lunar caves-system and the Elite Network Bavaria (ENB), Germany for providing funds for the academic program "Satellite Technology".

* Corresponding author.
E-mail address: fabian.arzberger@uni-wuerzburg.de (F. Arzberger).

https://doi.org/10.1016/j.robot.2024.104852

Available online 21 November 2024

Fig. 1. Two spherical mobile mapping systems, equipped with inertial sensors, cameras, and LiDAR. A high-resolution video where the filters run onboard and are compared to each other in real-time is available at https://youtu.be/oO93Y4nZ5w0. (Left spherical system:) The new prototype, featuring a 'Hesai Pandar-XT32' 360° horizontal FoV LiDAR with 0 cm minimum scanning distance. (Right spherical system:) The old prototype from previous work [1], featuring a 'Livox Mid-100' limited 98.4° horizontal FoV LiDAR with 100 cm minimum scanning distance.

errors grow exponentially with increasing distance. While working with spherical robots, non-centered rotation is the main movement of the internal sensors, which proposes a huge challenge to state of the art SLAM algorithms. In previous work, we proposed initial laser-based offline-SLAM solutions for simplified sub-problems, i.e., rotation while descending [13], and rolling on flat surfaces [14]. Our previously proposed Delta-filter [1] – on which this paper builds upon – did not make use of LiDAR measurements for pose estimation. Similarly, in this work we also address only the localization of the system and do not perform laser-based SLAM, but only use the LiDAR data for accuracy evaluation. However, in the near future, the output of our developed trajectory filters should be utilized as an initial guess for a laser-based online-SLAM system, yet this is beyond the scope of this work. The contributions of this work are as follows:

- A Kalman-filter design for pose estimation via sensor fusion on spherical mobile mapping systems.
- Comparison of the previously proposed Delta-filter [1] with the Kalman-filter.
- A second spherical prototype running both filters, using different hardware and sensor configuration compared to the previous paper [1].

The paper is structured as follows: In the next section, we provide an overview of state of the art 6-DoF pose filters, and outline the most similar approaches. Then, we introduce the "Delta"-filter in a general fashion and show an example implementation on a spherical mobile mapping system. Finally, we introduce our accuracy measures and experiments and show that the filter is able to deal with slow and fast motion as well as driving curves.

2. State-of-the-art

Many onboard multi-sensor pose estimation approaches exist in the community. The majority of which are implemented and developed towards autonomous driving cars [15,16], and unmanned aerial vehicles (UAV) [17,18]. Soloviev et al. [19] give a broad outline on the sensor types used for navigation: They define a self-contained inertial navigation system (INS) as the primary sensor, as it is available on any platform. Further, the authors consider the following secondary sensors which are qualified for fusion with the INS solution: Global Navigation Satellite System (GNSS) based (e.g. GPS), feature based (e.g. cameras or LiDAR), beacon based (e.g. using specialized navigation signals), or based on signals of opportunity (SoOP) (e.g. radio-frequency signals). In this work we will focus on visual-inertial navigation systems (VINS) and later propose a filter for our spherical system. Santoso et al. [20] categorize popular filters in the robotics community: (1) The Kalman Filter (KF) [21] has been designed to estimate the most likely system state under Gaussian noise by minimizing the covariances of the estimation error. It has since been reinvented and extended several times, leading to variants such as the Unscented Kalman Filter (UKF) [22], Extended Kalman Filter (EKF) [23], or Multistate Constrained Kalman Filter (MSCKF) [24], just to name a few. KF-based approaches are by far the most popular state estimators among the robotics community. Example implementations on different systems include [18,25–29]. (2) The H_∞ filter approach originates from control theory where it is used as an optimal robust controller. Instead of minimizing the covariance of the estimation error, the H_∞ filter minimizes the worst-case estimation error, which leads to better performance if modeling uncertainties are present [30]. (3) Particle filters (PF), or Monte-Carlo Methods, are known for being applied in many stochastic estimation problems [31]. By now, it is well-known that PF outperforms KF in nonlinear systems underlying non-Gaussian noise [32]. Its biggest drawback is the computational load required for processing many particles representing a single state. (4) Rao–Blackwellized Particle filters (RBPF) combine the advantages of PF and KF while getting rid of their major issues [33]. Therefore, if the system state model contains linear parts with Gaussian noise, these components are separated and processed using KFs, while nonlinear parts with non-Gaussian noise are dealt with PFs. And finally, in recent years we have noticed the use of (5) graph optimization based methods such as GOMSF [34] and VIRAL-Fusion [35], where the system states are represented and optimized in a pose-graph.

The above mentioned examples solely treat filters implemented on ground vehicles or UAV. Yet other examples exist that implement multi-sensor pose estimation on more challenging systems. Kim et al. [36] fuse data from four sensing modalities on an unmanned underwater vehicle (UUV) using an approach using covariance intersection based on nonlinear optimization. They consider measurements taken via acoustic ultra-short baseline (USBL), Differential GPS (DGPS), Doppler Velocity Logs (DVL), and an INS. Fang et al. [37] use three different sensors for pose estimation on wearable augmented reality (WAR): a monocular camera, a depth sensor, and an INS. They use a KF-based approach in a sliding window fashion. To our knowledge there exists only one onboard pose estimation filter for spherical robots [38]. This approach [38] comes from our own lab and uses only data from inertial measurement units (IMU). The basic idea is to combine the well known IMU orientation filters: the Madgwick filter [39] and Complementary filter [40]. As for translation, the filter performs dead-reckoning using the motion model of a rolling sphere and adding constraints for slipping and sliding effects. Furthermore, the output of the filter in [38] is being utilized as input for the filter proposed in this paper. Another niche filter, the EKF for spherical robots presented in [41], is similar to [38], as it performs attitude estimation and dead reckoning based only on an IMU. Lastly, we want to mention another filter that is much simpler than any of the approaches stated above, yet surprisingly effective: Gyrodometry [42]. This filter has been implemented to combine data from wheel encoders (Odometry) with data from a gyroscope by considering not the measured state, but instead the change of state. Therefore, the filter considers the similarity of the measurements to each other to eliminate outliers and update the current state accordingly, without using any covariance estimation. The previously proposed Delta-filter [1] is similar in these two aspects (change of state and similarity of measurements), but extends the idea to an arbitrary number of estimators in 6-DoF and adds a motion model. Our proposed Kalman-filter also considers the change of the state, however we add uncertainty modeling to our sensors in order to estimate the covariance matrices.

3. Sensor fusion with 6-DoF delta-filter

In this section we introduce common notation and reiterate the Delta-filter design from our previous work [1]. The Delta-filters purpose is to receive 6-DoF trajectory estimates from multiple sources, which are known to be unreliable, and filter them in a probabilistic way. We consider a trajectory "unreliable" if it accumulates drift or makes sudden jumps — which are common effects in IMU- and VIO-based estimators. The filtered trajectory does not utilize covariance estimations, does not use any information from future measurements, and is computed in real-time. However, similar to a Kalman filter, the Delta-filter requires a motion model, which is also considered unreliable. In our implementation we filter only two trajectory estimates with a given motion model, yet the Delta-filter is theoretically suitable for an arbitrary number of estimators.

3.1. Notation and filter design

Suppose we have multiple 6-DoF pose estimators $\boldsymbol{X} = [\boldsymbol{R}, \boldsymbol{p}]^\intercal \in \mathrm{SE}(3)$, where $\boldsymbol{R}$ is a 3×3 rotation matrix and $\boldsymbol{p}$ is a vector in $\mathbb{R}^3$. The pose of the kth estimator at time t is denoted by $\boldsymbol{X}_k(t) = \left[\boldsymbol{R}_k(t), \boldsymbol{p}_k(t)\right]^\intercal : \mathbb{R} \to \mathrm{SE}(3)$. Note that all poses from all estimators must first be transferred in a shared global coordinate frame. As the poses arrive at different time stamps, it is necessary to interpolate between measurements to capture all estimates at the same point in time. Thus, the Delta-filter computes an estimate at the rate of the slowest estimator, denoted as $\boldsymbol{X}_0$, yielding a query time t_q. We call the resulting pose $\boldsymbol{X}_0(t_q)$ the "measurement". All other estimators $\boldsymbol{X}_k$ are queried at time t_q, by interpolating between two measurements at given timestamps $t_{q\pm1}$, as shown in Fig. 2. Note that rotation matrices and unit quaternions are isomorphic, thus we use $\boldsymbol{q}_k(t)$ and $\boldsymbol{R}_k(t)$ interchangeably as they represent the same elements in SO(3). Then, the interpolation is constructed using quaternion spherical linear interpolation (Slerp) and linear vector interpolation as described by Eqs. (1) - (5):

$$\boldsymbol{X}_k(t_q) = \left[\boldsymbol{R}_k(t_q), \boldsymbol{p}_k(t_q)\right]^\intercal , \tag{1}$$

$$\hat{t} = \frac{t_q - t_{q+1}}{t_{q-1} - t_{q+1}} \in [0;1] , \tag{2}$$

$$\Omega = \cos^{-1}\left(\boldsymbol{q}_k(t_{q-1}) \cdot \boldsymbol{q}_k(t_{q+1})\right) , \tag{3}$$

$$\begin{aligned}\boldsymbol{R}_k(t_q) &= \mathrm{Slerp}\left(\boldsymbol{q}_k(t_{q-1}), \boldsymbol{q}_k(t_{q+1}), \hat{t}\right) \\ &= \frac{\sin((1-\hat{t})\Omega)}{\sin(\Omega)} \cdot \boldsymbol{q}_k(t_{q-1}) + \frac{\sin(\hat{t}\Omega)}{\sin(\Omega)} \cdot \boldsymbol{q}_k(t_{q+1}),\end{aligned} \tag{4}$$

$$\boldsymbol{p}_k(t_q) = (1-\hat{t}) \cdot \boldsymbol{p}_k(t_{q-1}) + \hat{t} \cdot \boldsymbol{p}_k(t_{q+1}) \tag{5}$$

The idea of the Delta-filter is to track the changes between given timestamps t_1 and t_2 (also known as "deltas") of the measurements and interpolations

$$\Delta\boldsymbol{X} = \left[\boldsymbol{R}^{-1}(t_2) \cdot \boldsymbol{R}(t_1) \,,\; \boldsymbol{p}(t_2) - \boldsymbol{p}(t_1)\right]^\intercal \tag{6}$$

and estimate a new delta that is more meaningful. That is to say that the Delta-filter estimates the most likely pose change between given timestamps. Therefore, the filter first estimates a model delta

$$\begin{aligned}\Delta\boldsymbol{X}_m &= \left[\Delta\boldsymbol{R}_m, \Delta\boldsymbol{p}_m\right]^\intercal \\ &= f\left(\Delta\boldsymbol{X}_0 \,,\; \{\Delta\boldsymbol{X}_k : k \in \mathbb{N}\}\right)\end{aligned} \tag{7}$$

where f denotes the motion model that estimates the true motion given the measured and interpolated deltas, $\Delta\boldsymbol{X}_0$ and $\Delta\boldsymbol{X}_k$. In a later section we will give an example for the motion model f when implementing the filter on a spherical robot.

3.1.1. Measurement, interpolation, and model

The measurement, interpolation, and model deltas $\Delta\boldsymbol{X}_0$, $\Delta\boldsymbol{X}_k$, and $\Delta\boldsymbol{X}_m$ respectively, are all considered unreliable. They are used to estimate the filtered pose $\boldsymbol{X}_e(t_j)$ by iteratively applying an estimated filtered delta $\Delta\boldsymbol{X}_e$ that happened between t_{j-1} and t_j:

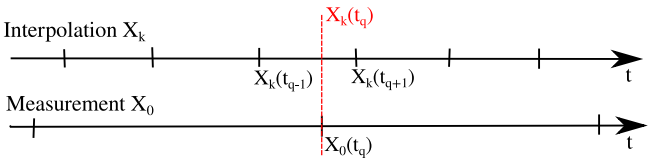


Fig. 2. Timelines showing two sensors publishing pose data at different rates. The sensor having the slower rate is defined as the "measurement", the other trajectories $\boldsymbol{X}_k$ get interpolated at measurement time t_q.

$$\boldsymbol{X}_e(t_j) = \Delta\boldsymbol{X}_e \cdot \boldsymbol{X}_e(t_{j-1}) \tag{8}$$

$$= \left[\Delta\boldsymbol{R}_e \cdot \boldsymbol{R}_e(t_{j-1}), \Delta\boldsymbol{p}_e + \boldsymbol{p}_e(t_{j-1})\right]^\intercal . \tag{9}$$

We separate the rotation and translation parts by assuming that the measured and interpolated orientations are sufficiently reliable estimates, i.e., they do not drift or jump between two consecutive frames such that averaging both rotation-deltas with Slerp works. Thus, we set the parameter $\hat{t}$ of the Slerp function to 0.5. To obtain the estimated filtered rotation delta $\Delta\boldsymbol{R}_e$, we compute

$$\Delta\boldsymbol{R}_e = \mathrm{Slerp}\left(\Delta\boldsymbol{q}_0, \Delta\boldsymbol{q}_k, \frac{1}{2}\right) \tag{10}$$

Note that for more than two estimators, the Slerp in Eq. (10) must be replaced with a different quaternion average, e.g. [43]. Furthermore, we assume that the estimated translation deltas $\Delta\boldsymbol{p}_0$, $\Delta\boldsymbol{p}_k$, and $\Delta\boldsymbol{p}_m$ are not sufficiently reliable to just average them, as inertial-tracking tends to drift and visual-tracking tends to jump.

3.1.2. Probabilistically weighted geometric mean

We calculate weights β_i for each delta that correspond to the similarity of the deltas distance to their geometric mean, thus outliers get a damped weight while similar values get a higher weighting:

$$|\hat{\boldsymbol{p}}| = \left(\prod_{i=1}^{n} |\Delta\boldsymbol{p}_i|\right)^{n^{-1}} , \tag{11}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n}\left(|\Delta\boldsymbol{p}_i| - |\hat{\boldsymbol{p}}|\right)^2} , \tag{12}$$

$$\beta_i = 1 - s^{-1} \cdot \left(|\Delta\boldsymbol{p}_i| - |\hat{\boldsymbol{p}}|\right) , \tag{13}$$

where $\Delta\boldsymbol{p}_i$ refers to the measurement, interpolation, and model deltas. Now we use these weights, corresponding to similarity, to average the translation direction and then scale it:

$$\Delta\boldsymbol{p}_e = \frac{d}{|\sum_i \beta_i \Delta\boldsymbol{p}_i|} \cdot \sum_i \beta_i \Delta\boldsymbol{p}_i , \tag{14}$$

where d is an estimate of the true scale of the translated distance:

$$d = \left(\prod_{\beta_i} |\Delta\boldsymbol{p}_i|^{\beta_i}\right)^{(\sum_i \beta_i)^{-1}} \tag{15}$$

3.2. Implementation on our spherical systems

Fig. 3 shows the reference frames of the sensors in a schematic drawing. We use three "PhidgetSpatial Precision 3/3/3 High Resolution 1044" [44] IMUs and an "Intel RealSense T265 stereo tracking camera" [45]. Our own IMU filtering algorithm [46] takes the 250 Hz raw outputs from the three onboard IMUs and computes a single pose estimate at 125 Hz. The fisheye cameras on the Intel T265 produce images at only 30 Hz. However, the T265 outputs a pose at 200 Hz using its own IMU. The implementation on our spherical systems uses two estimators, the IMU operating at 125 Hz defines the measurement $\boldsymbol{X}_0$, and the camera operating at 200 Hz defines the interpolation $\boldsymbol{X}_k$.

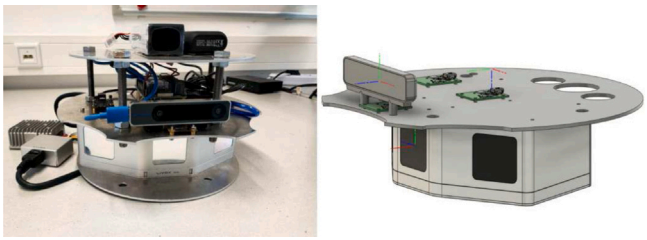


Fig. 3. (Left:) Spherical mobile mapping system without its protecting shell. (Right:) CAD model of the spherical system with sensor frames.

For the motion model f of the spherical system with known radius $r = 0.145$ m, we assume that rotation leads to translation, thus we calculate the estimated model delta using the arc length of rotation:

$$f(\Delta \boldsymbol{X}_0, \Delta \boldsymbol{X}_k) = \left[\Delta \boldsymbol{R}_e,\ r \cdot \angle\left(\Delta \boldsymbol{R}_e\right) \cdot \frac{\Delta \boldsymbol{p}_0 + \Delta \boldsymbol{p}_k}{|\Delta \boldsymbol{p}_0 + \Delta \boldsymbol{p}_k|}\right]^\intercal, \tag{16}$$

where $\angle(\cdot)$ denotes the angle around the axis described by the rotation matrix. Note that we just defined the model rotation $\Delta \boldsymbol{R}_m$ from to be equal to $\Delta \boldsymbol{R}_e$ from Eq. (10), as the orientation estimation is considered sufficiently reliable.

The simplicity of the filter design allows for the introduction of simple but effective design choices. For example, we notice that the yaw estimations of the IMUs tend to drift without the use of their magnetometers, whereas the tracking camera does not have this issue. This effect is also visible in the upcoming results by comparing the right columns of Figs. 5(a) with 5(b). Note that we also tried avoiding to rotate over the camera, such that it experiences less pitching motion but more roll, seen from the cameras reference frame. This way, the tracking camera is able to see the floor at all times, which helps with the sudden position jumps. However, this makes the yaw estimations of the camera much worse, presumably because it is not designed to handle strong self-rolling motions.

We show this effect in Fig. 4. Note that a potential fix might be including a second camera, mounted orthogonally to the other one, and including it in the filter as an additional estimator. However, we will address this in future research and stick to pitching the camera instead in this work. Due to the background of our spherical system, we do not want to use the magnetometers by design. Thus, we must rely more on the camera estimations for the yaw angle, which is why we exchange the estimation of the rotation delta in Eq. (10). Instead of only using Slerp, which is more universal, we first use Slerp and then replace the yaw-part of the resulting delta with the interpolated camera yaw delta. Hence, the change in yaw is only estimated via the camera.

4. Sensor fusion with Kalman-filter

The Kalman-filter (KF) we introduce here, similar to the Delta-filter, also explicitly filters the change of the system state, instead of the state itself. The state vector of the KF is then used to iteratively update the transformation $\boldsymbol{X}_e$ according to Eq. (9). In this section we utilize the notation introduced in the previous section, resulting in a state vector $\boldsymbol{y}_k$

$$\boldsymbol{y}_k = \begin{bmatrix}\Delta \boldsymbol{p}_k \\ \Delta \boldsymbol{q}_k \\ \boldsymbol{\omega}_k\end{bmatrix} = \begin{bmatrix}\left(\Delta p_k^x, \Delta p_k^y, \Delta p_k^z\right)^\intercal \\ \left(\Delta q_k^x, \Delta q_k^y \Delta q_k^z, \Delta q_k^w\right)^\intercal \\ \left(\omega_k^x, \omega_k^y, \omega_k^z\right)^\intercal\end{bmatrix} \tag{17}$$

where $\Delta \boldsymbol{p}_k$ is the position delta and $\Delta \boldsymbol{q}_k$ is the rotation delta as a unit quaternion that are used to update $\boldsymbol{X}_e$, $\boldsymbol{\omega}_k$ is the bias corrected angular velocity in a fixed global coordinate system. In the following, $\boldsymbol{n}$ denotes the normal vector of the ground that the sphere is rolling on without slipping. Note that $\boldsymbol{n}_k$ is theoretically measurable via the onboard LiDAR, yet this is beyond the scope of this work, which is why we assume a flat floor using a constant $\boldsymbol{n} = \boldsymbol{n}_k = (0, 0, 1)^\intercal$. The key observation here is that the velocity of the balls center $\boldsymbol{v}$, when rolling without slipping over the ground with radius r must be

$$\boldsymbol{v} = r\boldsymbol{\omega} \times \boldsymbol{n}\ , \tag{18}$$

where $\times$ denotes a vector cross product. We assume constant velocity between two consecutive frames, thus

$$\begin{aligned} &\Delta \boldsymbol{p} \sim \boldsymbol{v} \Delta t \\ \Rightarrow\ &\Delta \boldsymbol{p}_k \approx \left(r\boldsymbol{\omega}_k \times \boldsymbol{n}_k\right) \Delta t \\ &= r \begin{bmatrix} w_k^y n_k^z - w_k^z n_k^y \\ w_k^z n_k^x - w_k^x n_k^z \\ w_k^x n_k^y - w_k^y n_k^x \end{bmatrix} \Delta t\ . \end{aligned} \tag{19}$$

We use (19) to form a linear motion model. In order to minimize the effect of integrating errors over time due to the constant velocity assumption, the frequency of the measurements should be as high as possible, i.e., Δt should be as small as possible. It is possible to write the cross product using a skew-symmetric matrix $\boldsymbol{n}_\times$, which means we are able to abuse notation such that the resulting matrices in the Kalman-filter are more compact:

$$\left(r\boldsymbol{\omega}_k \times \boldsymbol{n}_k\right) \Delta t = \begin{bmatrix} 0 & n_k^z & n_k^y \\ -n_k^z & 0 & n_k^x \\ -n_k^y & -n_k^x & 0 \end{bmatrix} \begin{bmatrix}\omega_k^x \\ \omega_k^y \\ \omega_k^z\end{bmatrix} r \Delta t \tag{20}$$

$$= \boldsymbol{n}_\times^\intercal \boldsymbol{\omega}_k r \Delta t \tag{21}$$

Using the assumption from the previous section we omit to predict the rotation change $\boldsymbol{q}_k$, but rather weight the individual measurements via the later described covariance model, which leads to a formulation of the Kalman-filter prediction in linear form (LKF):

$$\hat{\boldsymbol{y}}_k = \boldsymbol{F}_k \boldsymbol{y}_{k-1} + \boldsymbol{G}_k \boldsymbol{u}_k \tag{22}$$

$$\hat{\boldsymbol{P}}_k = \boldsymbol{F}_k \boldsymbol{P}_{k-1} \boldsymbol{F}_k^\intercal + \boldsymbol{Q}_k \tag{23}$$

$$\hat{\boldsymbol{z}}_k = \boldsymbol{H}_k \hat{\boldsymbol{y}}_k\ . \tag{24}$$

The matrices $\boldsymbol{F}_k$, $\boldsymbol{G}_k$, and $\boldsymbol{H}_k$, considering that the system input is the angular velocity ($\boldsymbol{u}_k = \boldsymbol{w}_k$), each state entry gets measured ($\hat{\boldsymbol{z}}_k = \hat{\boldsymbol{y}}_k$), and the motion model from Eq. (19), consequently are:

$$\boldsymbol{F}_k = \begin{bmatrix} \boldsymbol{0}_{3\times7} & r\Delta t \boldsymbol{n}_\times^\intercal \\ \boldsymbol{0}_{4\times3} & \boldsymbol{I}_{4\times3} \quad \boldsymbol{0}_{4\times3} \\ & \boldsymbol{0}_{3\times10} \end{bmatrix}, \tag{25}$$

$$\boldsymbol{G}_k = \begin{bmatrix} \boldsymbol{0}_{7\times10} \\ \boldsymbol{0}_{3\times7} \quad \boldsymbol{I}_{3\times3} \end{bmatrix}, \tag{26}$$

and

$$\boldsymbol{H}_k = \boldsymbol{I}_{10\times10}. \tag{27}$$

The update step of the Kalman-filter is as usual:

$$\boldsymbol{S}_k = \boldsymbol{H}_k \hat{\boldsymbol{P}}_k \boldsymbol{H}_k^\intercal + \boldsymbol{R}_k \tag{28}$$

$$\boldsymbol{K}_k = \hat{\boldsymbol{P}}_k \boldsymbol{H}_k^\intercal \boldsymbol{S}_k^{-1} \tag{29}$$

$$\boldsymbol{y}_k = \hat{\boldsymbol{y}}_k + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k\right) \tag{30}$$

$$\boldsymbol{P}_k = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k\right) \hat{\boldsymbol{P}}_k \tag{31}$$

This update step (Eqs. (28)–(31)) is performed for each estimator—twice in our case, where $\boldsymbol{z}_k$ contains the respective sensors measurements.

4.1. Covariance model on spherical system

Many sensors (e.g., Intel's D-series tracking cameras, or recent PhidgetSpatial IMU devices) have a built-in function to supply covariance matrices. If this is available, we recommend using these estimations. Otherwise, we now introduce a model for prior covariance estimation based on the following simple heuristics:

Fig. 4. Resulting 3D point cloud when using only the pose estimations of the tracking camera as frames. Instead of pitching the camera, such that it looses track when looking on the floor, here the camera primarily experienced roll. (Left:) Sliced birds-eye view of the point cloud. Yaw estimations suffer when rolling the camera instead of pitching. (Right:) Sliced side view of the point cloud. No sudden jumps are visible when rolling the camera instead of pitching.

- (1) High angular velocities lead to a loss in accuracy for all estimators.
- (2) When the camera looses track, e.g., when looking at the floor, or unfavorable lighting conditions, its accuracy suffers.
- (3) The IMU's do not use their magnetometers by design, thus, estimation of the yaw angle should prefer camera measurements.

Therefore, we initialize the measurement noise matrices of the camera $\boldsymbol{R}_{0,\mathrm{CAM}}$ and IMU's $\boldsymbol{R}_{0,\mathrm{IMU}}$ on the diagonal using a pre-calculated precision of the sensors when the system does not move, then add 10% to the each value. In order to fulfill the third heuristic, the parts responsible for yaw estimation in $\boldsymbol{R}_{0,\mathrm{IMU}}$ get multiplied by a factor of 100. Then, we calculate the measurement noise matrices used in the Kalman-filter at each step k by scaling the initial covariance matrices exponentially with rotational velocity to account for the first heuristic:

$$\boldsymbol{R}_{k,\mathrm{IMU}} = e^{|\boldsymbol{\omega}_k|} \cdot \boldsymbol{R}_{0,\mathrm{IMU}} \tag{32}$$

$$\boldsymbol{R}_{k,\mathrm{CAM}} = e^{|\boldsymbol{\omega}_k|} \cdot B_k \cdot \boldsymbol{R}_{0,\mathrm{CAM}} \tag{33}$$

The scalar factor B_k corresponds to an internal confidence variable of the camera tracking module, which ranges from 1 (high feature-tracking confidence) to 1000 (failed to track any features) to account for the second heuristic.

5. Experiments and evaluation

In our previous work we compared the Delta-filter to the baselines. For completeness, these results will also appear in the following subsections. Fig. 5 qualitatively summarizes the results by showing the resulting point-clouds when applying the trajectories of each estimator directly to the LiDAR data. The IMU-based approach (a) suffers from drift in the yaw axis and overestimates the scale of the trajectory. The visual-inertial (b) tracking approach tends to jump whenever the camera looses track, which happens quite often given the unfavorable type of sensor motion. Our previously proposed Delta-filter (c) combines both trajectories, gets rid of the drift and jumps, and estimates the scale of the trajectory better. Additionally, we test our spherical system with another state-of-the-art visual-inertial navigation system (VINS): "VINS-Fusion" [47,48]. It is an extension of "VINS-Mono" [49] to support multiple sensors, e.g., in our case a stereo camera setup with an IMU. We provide the calibration parameters and configuration files needed to run VINS-Fusion with the Intel T265 tracking camera on our github [50]. Fig. 6 shows that the VINS estimator diverges as soon as the rolling motion of the spherical system starts.

In this work, we have also made additional experiments to compare the Kalman-filter to the Delta-filter. Qualitatively speaking, the first row of Fig. 17 shows that the resulting point-clouds when applying the Kalman-filter estimations appear to have less drift. Furthermore, we apply a simple offline point-to-point ICP to the LiDAR data in a forward-pass fashion in order to simulate on-board processing, where no future measurements are available.

Fig. 7 shows the resulting point-cloud. The following sections quantify the results using ground truth trajectories and maps.

5.1. Error metrics

To quantify the quality of pose estimation, we use two principal approaches: On the one hand, we measure ground truth trajectories with an Optitrack system using IR reflectors. On the other hand, we also compare the resulting point clouds against ground truth measurements in larger environments, when Optitrack is no longer available. We denote the ground truth trajectory $\boldsymbol{X}_{\mathrm{ref}} = \left[\boldsymbol{R}_{\mathrm{ref}}, \boldsymbol{p}_{\mathrm{ref}}\right]^{\mathsf{T}}$, and the other estimated trajectories $\boldsymbol{X}_{\mathrm{est}} = \left[\boldsymbol{R}_{\mathrm{est}}, \boldsymbol{p}_{\mathrm{est}}\right]^{\mathsf{T}}$. For each timestamp in the ground truth trajectory, we sample the closest pose in time from the estimated trajectory for correspondence. Note that all the trajectories must be aligned with the ground truth trajectory. Therefore we align the origins of the trajectories first, as we know that all trajectories started from the same point. Afterwards we rotate around the shared origin using a least-squares alignment according to Umeyama [51]. Note that we only use the estimated rotation of the Umeyama method, since we already aligned the origins. From this point, we use Grupp's [52] software for trajectory evaluation. We record each individual LiDAR frame on-board using the Robot Operating System (ROS). The resulting point clouds are aligned to ground truth using the well-known Iterative Closest Points (ICP) algorithm. We use 3DTK [53] for the processing of the point clouds, which includes merging the individual frames into one scan, visualization, using ICP, matching to ground truth, and creating difference point-clouds with corresponding histograms and RMSE.

5.1.1. Absolute position error

The absolute position error (APE) represents the unsigned error of the translation estimation and is given by the magnitude of the difference vector of both translation vectors

$$\mathrm{APE}_i = |\boldsymbol{p}_{\mathrm{est},i} - \boldsymbol{p}_{\mathrm{ref},i}| \,. \tag{34}$$

5.1.2. Relative pose error

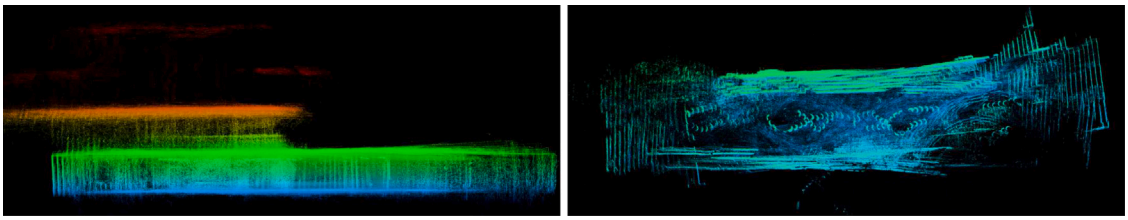
The relative pose error (RPE) represents the error of the orientation estimation by first calculating the orientation difference in rotation matrix form, and then extracting the unsigned angle from the corresponding angle-axis representation. It is thus given by

$$\mathrm{RPE}_i = \left| \angle \left(\left(\boldsymbol{R}_{\mathrm{ref},i}^{-1} \boldsymbol{R}_{\mathrm{ref},i-1} \right)^{-1} \left(\boldsymbol{R}_{\mathrm{est},i}^{-1} \boldsymbol{R}_{\mathrm{est},i-1} \right) \right) \right| \,. \tag{35}$$

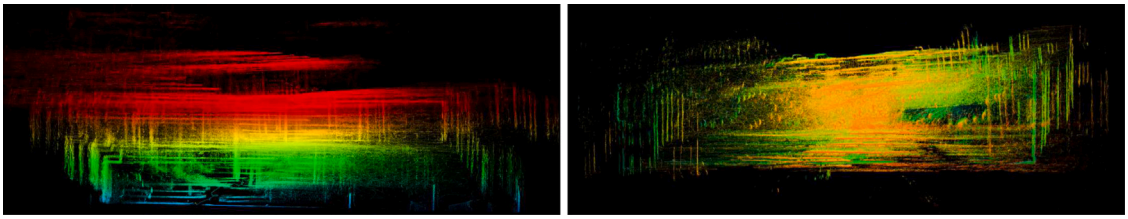
5.1.3. Point cloud error

The point cloud error represents the root of the mean squared point-to-point errors (RMSE). Suppose, after matching with ICP, there are N corresponding model- and data-points in the same coordinate frame, denoted $\boldsymbol{m}_i, \boldsymbol{d}_i \in \mathbb{R}^3$ respectively. Then, the root mean squared error is given by

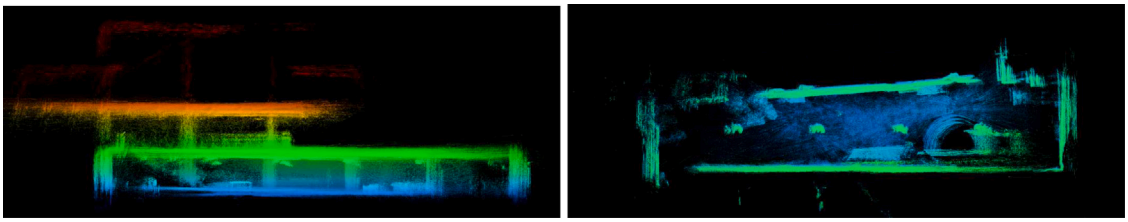
$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \sum_{i=0}^{N} |\boldsymbol{m}_i - \boldsymbol{d}_i|^2} \tag{36}$$



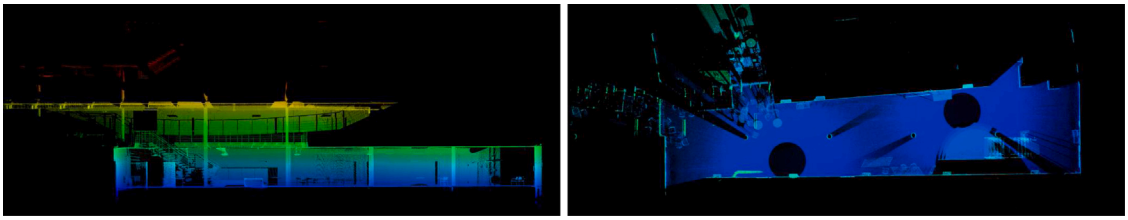
(a) Dead-reckoning IMU-only based pose estimation due to Zevering et al. [47]



(b) Visual-inertial tracking using an Intel RealSense T265 stereo-camera



(c) The Delta-filter using (a), (b), and a motion model



(d) Ground truth point cloud available from a RIEGL VZ-400 TLS

Fig. 5. Resulting point clouds when using three different estimators (a), (b), and (c) are orthographically visualized. A ground truth point cloud is shown in (d). Images in one column were shot from the same point of view. The colors of the points in the point cloud denote their height. Blue corresponds to lower, whereas red corresponds to larger height values. The left column shows sliced views from the side, whereas the right column shows sliced views from the birdseye perspective.

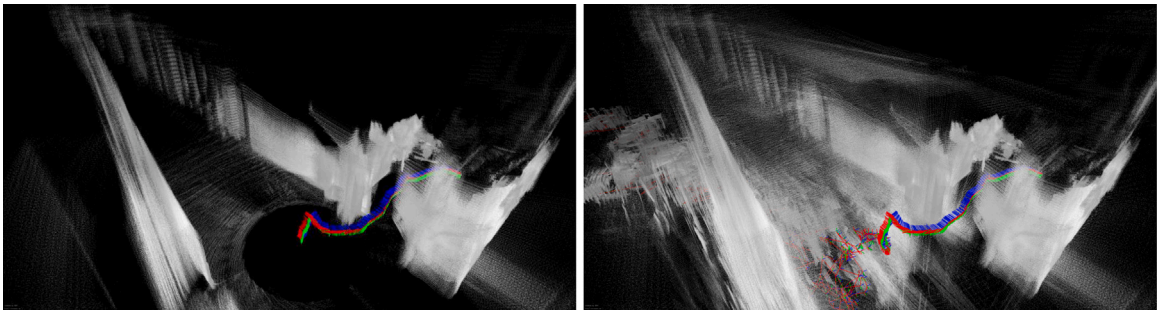


Fig. 6. Resulting point clouds and trajectory for VINS-Fusion [47,48]. (Left:) We carry the spherical system by hand without rolling. The resulting trajectory is consistent with the environment seen in the point cloud. (Right:) We place the spherical system on the floor and introduce rolling motion. The VINS estimator diverges. The resulting trajectory and point clouds are inconsistent and unusable.

5.2. Comparing delta-filter against baselines

The experiments consist of three types of motion: rolling a straight line slowly, fast, and driving curves at moderate speed. In the first two experiments, an OptiTrack system is available to capture ground truth trajectories, such that we are able to use Eqs. (35) and (34). However, in the last experiment (driving curves), the environment and trajectory is larger, making the OptiTrack system unavailable. In this experiment, we use a Riegl VZ-400 terrestrial laser-scanner (TLS) with an angular resolution of $0.04°$ and accuracy of $5\,\mathrm{mm}$ to provide accurate ground truth point clouds. As our system is equipped with a laser-scanner (see Fig. 3), we compare the resulting point cloud to the ground truth map using Eq. (36). Both setups are shown in Fig. 8.

5.2.1. Fast motion

In this experiment, the sphere traversed a distance of approx. $4\,\mathrm{m}$ in about $10\,\mathrm{s}$. Fig. 9 shows the APE (34) of all estimators over time. The T265 suffers from the highest error due to tracking loss, which forces it to rely solely on error prone double integration of acceleration measurements. The IMU-based approach show a considerable increase

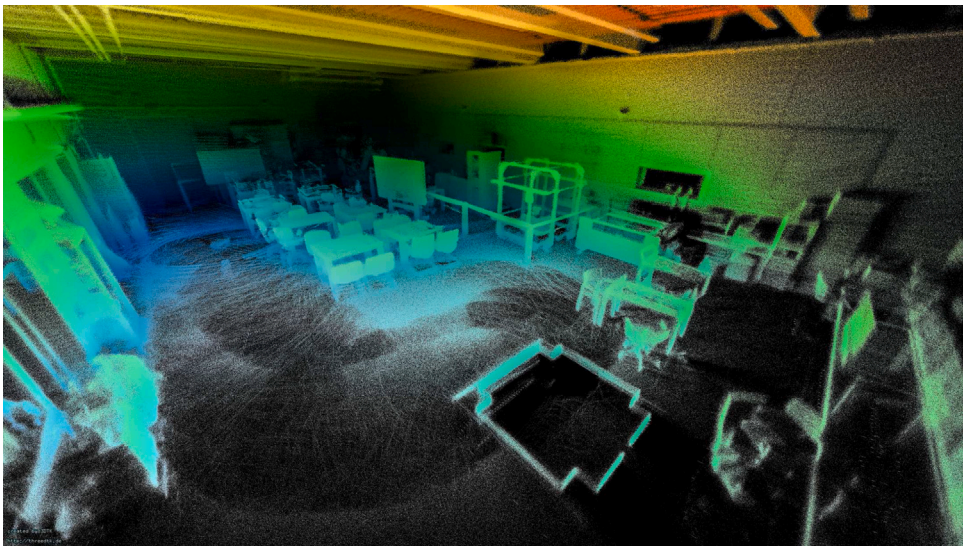


Fig. 7. The resulting point-cloud after using point-to-point ICP with the Kalman-filter estimated trajectory as initial guess. The color of the points in the point cloud denotes their height, where blue corresponds to lower and red to larger height values. The sphere got rolled manually by hand, however the operator is not visible in the point-cloud since we are only visualizing points with a minimum distance of 150 cm to the local origin of the scanner. A fly-through video of the point-cloud is available at https://youtu.be/oO93Y4nZ5w0.

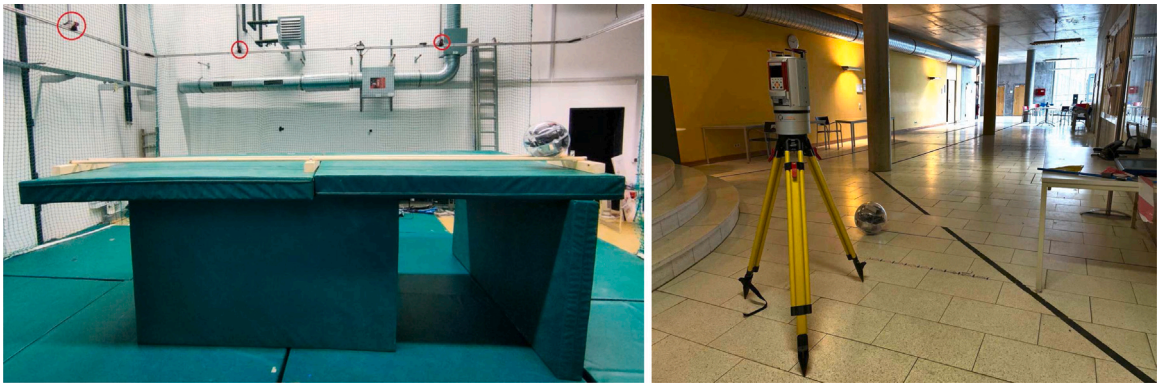


Fig. 8. (Left:) Laboratory test setup in a flycage equipped with an Optitrack system. The sphere has IR reflectors attached to its shell, which are detected by the cameras (red circles). (Right:) Laboratory test setup in the Computer Science building. A RIEGL VZ-400 TLS captures a precise ground truth point cloud for comparison with the spherical mobile mapping system. In both images, motion of the sphere is initiated manually by hand.

Table 1
Comparison of the estimated translation of the trajectory produced by the Delta-filter with its two source estimators, based on several statistical metrics. Each column compares three values where lower is better.

Error metrics to ground truth trajectories for fast and slow motion with respect to translation								
Estimator	RMSE [m]		Mean [m]		Std. [m]		Max. [m]	
	Slow	Fast	Slow	Fast	Slow	Fast	Slow	Fast
Dead-reckoning INS	1.713	1.736	1.447	1.291	0.917	1.160	2.882	3.001
Intel T265 Stereo-VIO	4.486	7.441	4.012	5.290	2.008	5.234	5.848	13.549
Delta-filter	**0.114**	**0.248**	**0.103**	**0.193**	**0.049**	**0.165**	**0.189**	**0.428**

Table 2
Comparison of the estimated rotation of the trajectory produced by the Delta-filter with its two source estimators, based on several statistical metrics. Each column compares three values where lower is better.

Error metrics to ground truth trajectories for fast and slow motion with respect to rotation								
Estimator	RMSE [deg]		Mean [deg]		Std. [deg]		Max. [deg]	
	Slow	Fast	Slow	Fast	Slow	Fast	Slow	Fast
Dead-reckoning INS	1.389	4.318	1.281	3.270	**0.537**	2.819	**2.653**	8.883
Intel T265 Stereo-VIO	**1.374**	**4.213**	**1.264**	**3.190**	0.541	**2.753**	2.701	**8.852**
Delta-filter	1.384	4.305	1.273	3.248	0.543	2.825	2.752	9.199

of error due to the accumulated drift. The error of the Delta-filter are orders of magnitude smaller compared to the IMUs and T265.

Fig. 10 shows the comparison of RPE (35) over time. Note that the Savgol-filter [54] is applied to the error signals. This is because

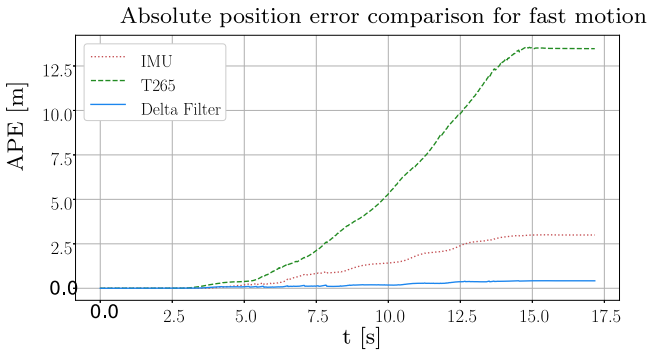


Fig. 9. The absolute position error of all estimators during fast motion over time.

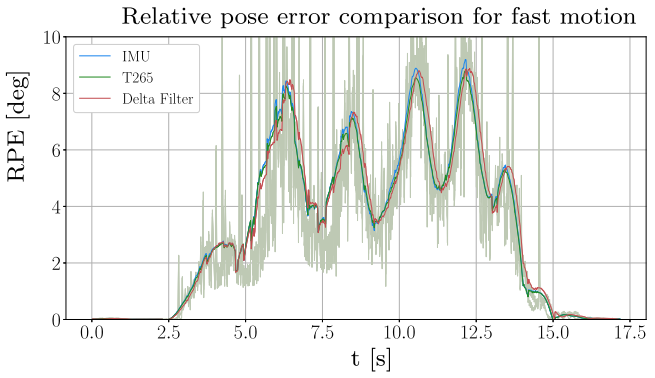


Fig. 10. The relative pose error of all estimators during fast motion over time. The Savgol-filter is applied with a window size of 51 and a polynomial degree of 3 to remove the effect of outliers. In the background the noisy pre-filtered data is shown with low opacity.

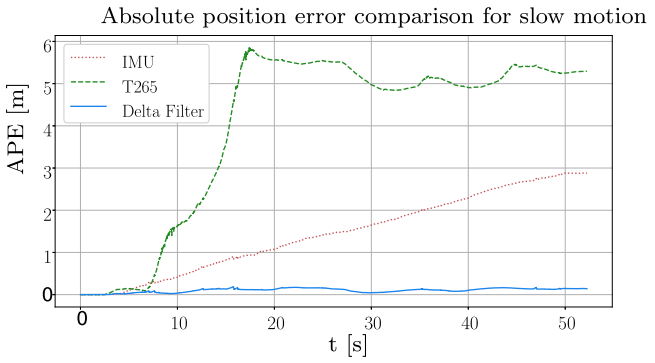


Fig. 11. The absolute position error of all estimators during slow motion over time.

the ground truth orientations from the OptiTrack system contain many outliers due to mirroring of the IR-reflectors on the spherical shell. The Savgol-filter removes the effect of these outliers but preserves the signal tendency. The RPE of all estimators do not differ particularly from each other, which is also evident from the error metrics in Table 2. In fact, the RMSE of the RPE of the Delta-filter is between the INS- and T265-solution, which makes sense considering the interpolation in Eq. (10).

5.2.2. Slow motion

In this experiment, the sphere traversed a distance of approx. 4 m in about 45 s. Fig. 11 shows the comparison of APE over time. The Delta-filter compensates for the linear accumulation of error of the IMU and the sudden jump of the T265, resulting in a lower overall translation error. Table 1 confirms this observation.

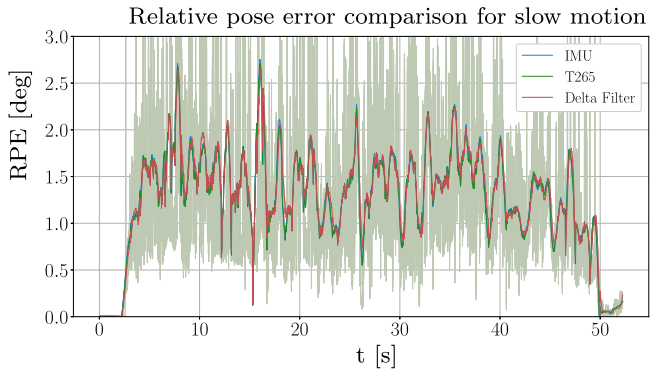


Fig. 12. The relative pose error of all estimators during slow motion over time. The Savgol-filter is applied with a window size of 51 and a polynomial degree of 3 to remove the effect of outliers. In the background the noisy pre-filtered errors are shown with low opacity.

Fig. 12 presents the comparison of RPE over time. As mentioned above, the Savgol-filter is applied on the error signals. The orientation errors of all estimators are similar to each other, yet overall smaller compared to fast motion.

5.2.3. Curves

Fig. 13 shows the result of the point cloud analysis. The error to ground truth is visualized in a point-to-point distance distribution histogram. Note that the large errors at the pillars are caused by global filter drift. On the other hand, the errors at the ceiling of the upper floor are rather caused by missing points in the ground truth. These points, however, are so few that they are only barely visible in the histogram. The mean point-to-point error from Eq. (36), which is our accuracy estimate for mapping, is 18.6 cm.

5.3. Comparison with Kalman-filter

In this section, we compare the previously evaluated Delta-filter with the Kalman-filter implemented in this work. For the evaluation, we use the same metrics as before and provide an analysis based on a ground-truth trajectory, as well as an analysis based on a ground truth map. In total, we perform two experiments: First, for the trajectory-based analysis we utilize the infrared marker tracking system “OptiTrack” available in the flycage again (see Fig. 8) to obtain the ground-truth trajectory. However, this time we tried rolling a circular loop instead of a straight line. Second, for the point-cloud based analysis we have the same RIEGL VZ-400 TLS available to create the ground-truth map, but perform the experiment in a different laboratory environment.

5.3.1. Trajectory-based analysis

Although posing additional challenges for the OptiTrack system due to marker occlusion and shell reflections, the trajectory this time is a circular loop. We filter these erroneous readings in the following way: First, we use the erroneous ground truth data to calculate the error metrics RPE and APE as before. Then, we use the earlier mentioned Savgol-filter [54] for smoothing. Note that filtering the data this way only does not remove outliers, but dampens their effect. However, a complete and more precise ground truth trajectory is most certainly desired, which is subject to future work. One approach might be to utilize the LiDAR measurements in a post-processing SLAM algorithm to obtain the exact trajectory of the sensor. Another solution for online estimation could be to implement a camera-vision based approach with visual markers, or change detection. In the experiment, the sphere traversed a circular loop with a distance of approx. 10 m in about 90 s, which is comparable to the speed of the previous slow motion

Table 3
Comparison of the estimated translation of the trajectory produced by the Delta-filter with the Kalman-filter, based on several statistical metrics. Each column compares two values where lower is better.

Statistics for APE and RPE error metrics to ground truth trajectories								
Estimator	Median		Mean		Std.		RMSE	
	APE [m]	RPE [°]	APE [m]	RPE [°]	APE [m]	RPE [°]	APE [m]	RPE [°]
DLIO	0.086	**0.879**	0.090	**2.132**	0.036	8.855	0.097	9.108
FASTLIO	**0.081**	0.926	**0.081**	2.168	**0.030**	**8.815**	**0.086**	**9.078**
Motion model	**0.081**	0.926	**0.081**	2.168	**0.030**	**8.815**	**0.086**	**9.078**

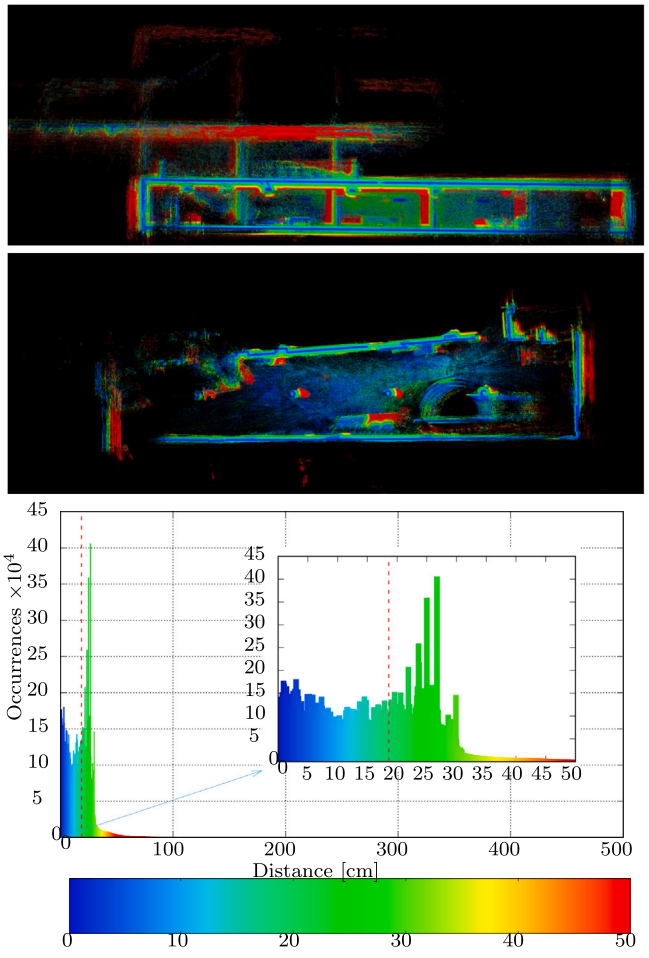


Fig. 13. Resulting point cloud (sliced side-view and birds-eye view) using the trajectory of the Delta-filter, as well as a histogram showing a distribution of point-to-point distances. These distances to ground truth are also visualized using color, where blue corresponds to lower and red to higher distances. The color spectrum at the bottom of the figure below the histrogram serves as a legend. The red dashed line in the histogram indicates the mean point-to-point error, which is 18.6 cm.

experiment. Fig. 14 visualizes the 3D path estimation of the Delta-filter (left) and the Kalman-filter (right), and compares it with ground truth. The color of the compared paths corresponds to the APE metric. Fig. 15 compares the same error metric, APE, as a time series. Fig. 16 shows the time series comparison of the RPE metric. In the time series, the effect of the IR-marker misinterpretations are clearly visible considering the unreasonable spikes for small durations, present in both the APE and RPE plots. Table 3 list more statistic regarding the error metrics, this time including the median value instead of the maximum value, considering the huge outliers. The Table 3 suggests that the rotation estimation of the Delta-filter outperformed the Kalman-filter, considering the median and mean values. However, this result is not significant considering the huge variance due to the outliers. Thus, we also compare the performance of the filters using a point-cloud-based analysis.

5.3.2. Point-cloud-based analysis

The experiment in this section is similar to the one from the previous point-cloud-based analysis. In a laboratory environment, rolling on a flat surface floor, the sphere traversed a distance of approx. 30 m in around 210 s. According to previously used term this is a slow motion to ensure a good mapping result, considering the next section where we utilize offline LiDAR-based SLAM. In this section, though, we do not use LiDAR-based SLAM, but apply the estimated trajectories of the Delta-filter and Kalman-filter directly to the LiDAR measurements, then match the resulting point-clouds against a ground truth map to evaluate the accuracy. Fig. 17 shows a direct comparison of the two maps. Each column represents one filter, where the left column corresponds to the Delta-filter and the right column to the Kalman-filter. The mean point-to-point error for the Delta-filter is 20.99 cm, and for the Kalman-filter it is 17.66 cm. The walls in the map created with the Kalman-filter align better than in the Delta-filter map, indicating that the Kalman-filter gives better estimates for the position of the system.

5.4. Offline mapping

Our new prototype (see left sphere in Fig. 1) is equipped with a 360° horizontal FOV, 0 cm minimum scanning–distance "Hesai Pandar-XT32" LiDAR operating at 20 Hz. Compared to the much smaller FOV and larger minimum scanning distance of the previous system ("Livox Mid-100"), the new setup is more suitable for mapping, as more points are available at all times. In this section we show which mapping result is possible, if using the trajectory estimation of the Kalman-filter as an initial guess for the well known offline point-to-point ICP, implemented in 3DTK [53]. Note that theoretically a better mapping result could be possible using a more sophisticated, globally consistent graph-based LiDAR SLAM algorithm, however this would not simulate LiDAR odometry. The ICP algorithm we use matches the current scan (not utilizing future measurements in order to simulate online mapping) against a metascan consisting of a window of past 200 matched scans. We do not skip any scan and downsample each one to include one point per 10 cm. Furthermore, we discard any point-to-point correspondences having a distance larger than 25 cm.

Fig. 18 shows the resulting maps, where each point is colored according to its distance to ground truth. The mean point-to-point error according to the histogram is 8.92 cm. Note that the red parts (indicating errors larger than 50 cm) are the result of lacking coverage in the ground truth map. Furthermore, the runtime of the ICP algorithm was 88 min on an "Intel i7-10750H" 12 core laptop CPU. This indicates that it is possible to run a trimmed online version of that ICP algorithm onboard for low-frequency LiDAR odometry. For example, one could decrease the sliding window size of the metascans, and not include every single LiDAR frame, but only a subset of keyframes. This will be subject to future work.

5.5. Discussion

In our previous paper we showed that the Delta-filter significantly improves the pose estimation accuracy, reduces drift, and eliminates

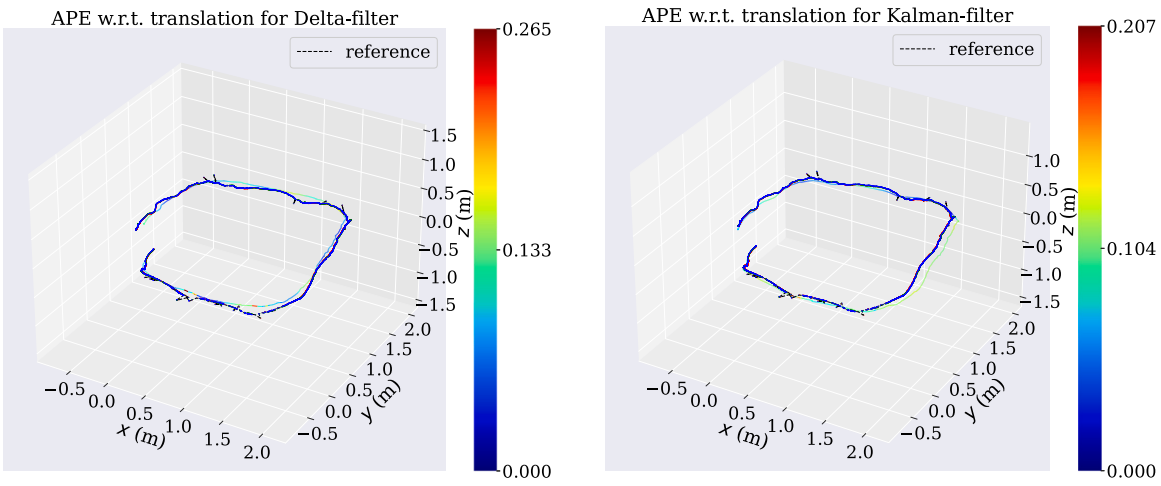


Fig. 14. Comparison of the absolute position error (APE) visualized as a 3D path. Spikes are present in the reference due to misinterpretation of IR-marker reflections off the spherical shell. The color of the compared trajectory denotes distance to ground truth. The color spectra to the right of the plots serve as a legend.

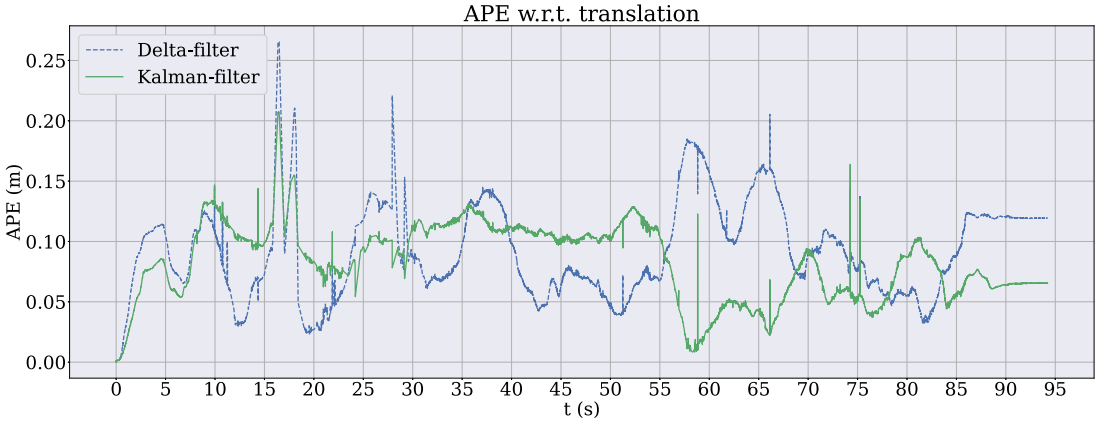


Fig. 15. Comparison of the absolute position error (APE) of the Delta- and Kalman-filter visualized as time series. Spikes appear due to erroneous ground truth caused by tracking the IR-marker reflections of the spherical shell.

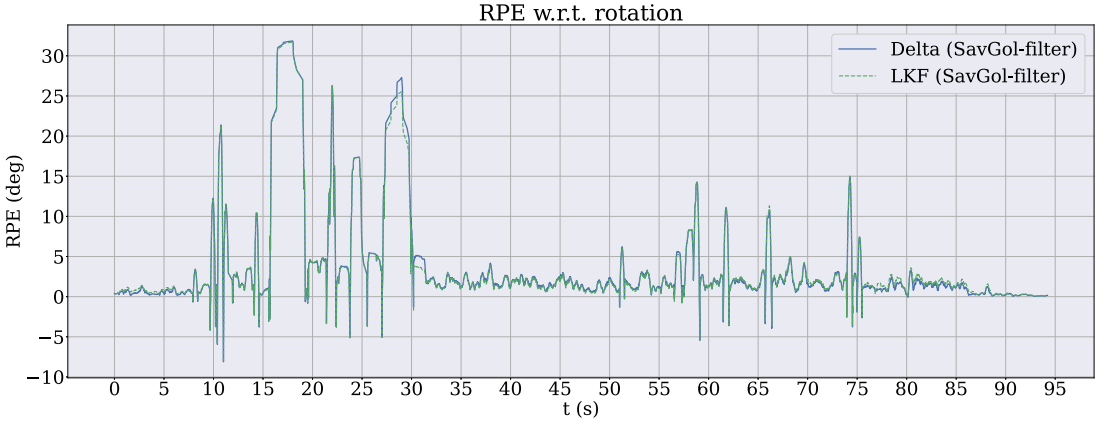
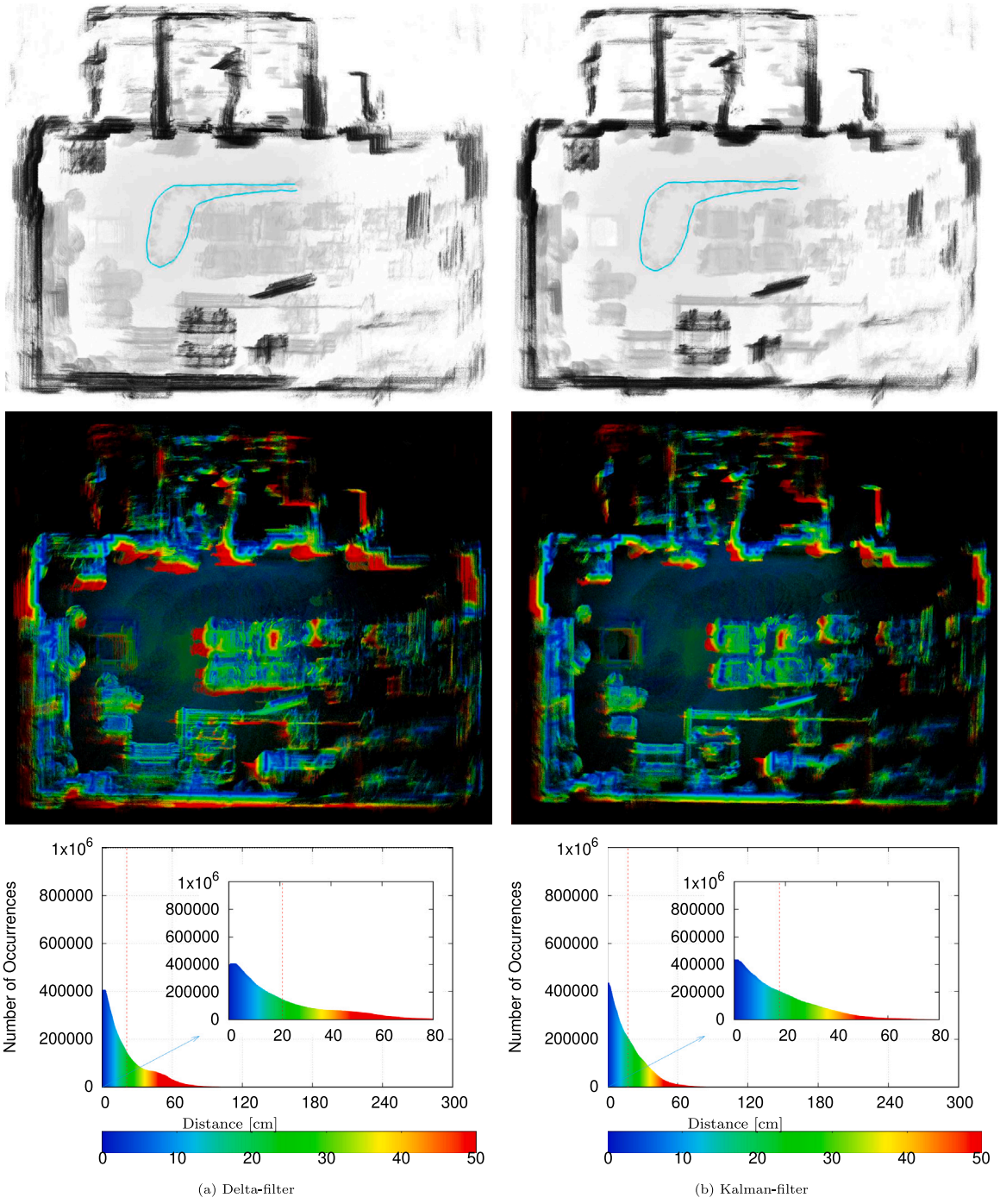


Fig. 16. Comparison of the relative pose error (RPE) of the Delta- and Kalman-filter visualized as time series. Spikes appear due to erroneous ground truth caused by tracking the IR-marker reflections of the spherical shell.

jumps on our spherical mobile mapping system. However, despite reducing the drift, all experiments showed that the Delta-filter still suffers from global drift regarding translation. Our analysis in this work shows that the Kalman-filter introduced in this work manages to reduce this drift even further. The Kalman- and Delta-filter run simultaneously onboard on an "Intel Gemini Lake N4120 Quad-Core 2.6 GHz" CPU where they introduce 10% mean load in total (20% peak) and drop no frames.

The offline mapping results, utilizing the Kalman-filter trajectory estimations, indicate that the drift could be completely compensated by



(a) Delta-filter (b) Kalman-filter

Fig. 17. Comparison of resulting point-clouds when applying the trajectory estimations of the filters to the LiDAR data. In the first row, the blue line represents the trajectory. In the second row, the points are colored according to their distance to ground truth. The histogram colors correspond to the colors of the points. The color spectrum at the bottom of the figure serves as a legend. The mean point-to-point error according to the histogram is 20.99 cm for the Delta-filter, and 17.66 cm for the Kalman-filter.

utilizing a quite simple direct point-to-point matching algorithm, such as ICP. It is too computationally expensive to utilize every individual scan onboard for matching. Thus, a better strategy, e.g., utilizing keyframes instead of each scan, is required to perform onboard real-time LiDAR SLAM. Furthermore, even after matching, the walls in the resulting point clouds appear to be thicker than in the ground truth point cloud, which comes down to two factors: First, the scanners used in the experiments have higher measurement noise, especially when the laser goes through the plastic shell. And second, the extrinsic calibration of the sensors in the spherical system is rather poor, as all the sensors assume to sit inside the center of the sphere. Also note that due to the challenges experienced using the infrared markers together with the reflective shell of the sphere, we will explore alternative methods of obtaining a ground-truth trajectory for these systems in future work. Possible solutions might be an approach based on change-detection using regular cameras, or utilizing more advanced offline post-processing of the LiDAR measurements to refine the trajectory. The

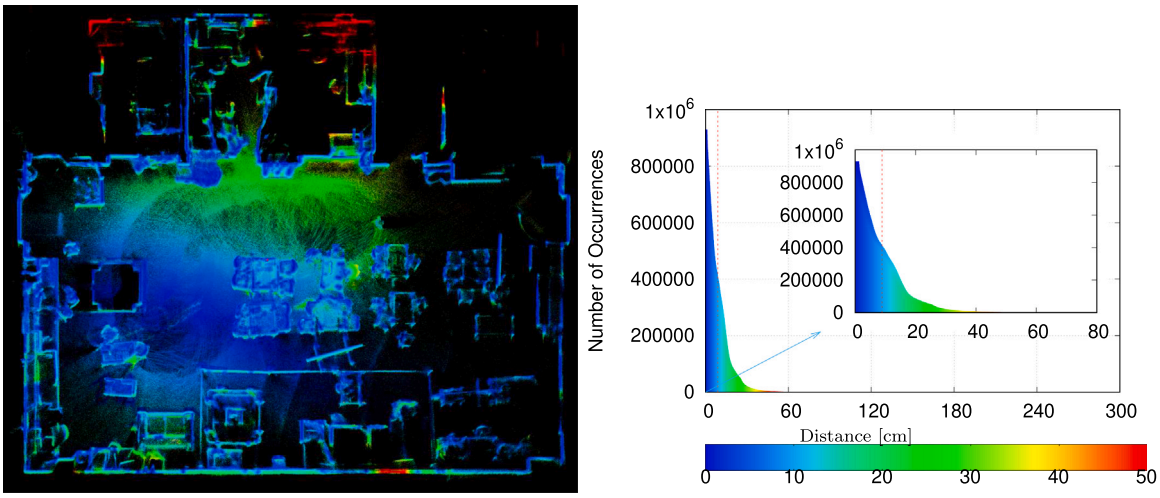


Fig. 18. Sliced birds-eye view of the resulting point-cloud after using point-to-point ICP with the Kalman-filter estimated trajectory as initial guess. The points are colored to represent their distance to ground truth. The color spectrum at the bottom of the histogram serves as a legend. The mean point-to-point error according to the histogram is 8.92 cm.

latter approach sound especially useful if the trajectory of the LiDAR is to be estimated, instead of the trajectory of the sphere center.

6. Conclusions

In this paper we addressed the problem of precise, real-time, and onboard localization in 6-DoF for spherical mobile mapping systems. Usually on these systems, the large angular velocities and constant aggressive dynamics when rolling makes state-of-the-art approaches, e.g. INS- or VIO-based solutions, more difficult. In our previous work, we proposed the simple yet effective Delta-filter, which is able to do real-time sensor fusion of an INS-with a VIO-based solution without using covariances on a spherical mobile mapping system. In this work, we refined our filter by implementing a Kalman-filter using heuristically estimated covariences. Our linear Kalman-filter design, similar to the Delta-filter, filters the change of the state, instead of the state itself. We implemented a linear motion model for rolling based on the cross-product of the angular velocity vector with the normal vector. The extensive analysis shows that these implementations have lead to improvements regarding the quality of the pose estimation on our spherical systems. Our new prototype (both are shown in Fig. 1) equipped with a laser-scanner which is better suited for spherical systems (360° horizontal FOV, 0 cm minimum scanning distance), combined with the refined Kalman-filter pose estimations, allowed us to create precise maps using a simple point-to-point ICP. Using this offline-ICP, we estimate the mapping accuracy of the spherical mobile mapping system to be 8.92 cm. The mapping process with ICP was forward-pass only, not utilizing future measurements, indicating that LiDAR odometry at a low frequency is possible. Thus, implementing online LiDAR odometry will be the next step in our future work to further improve the pose estimation and mapping quality, working towards a fully autonomous online SLAM. However, needlessly to say, a lot of work remains to be done. In the future, we need to address a proper extrinsic calibration for the sensors with respect to the spheres center to further increase the accuracy. This will be the foundation for an accurate motion distortion compensation algorithm for spherical systems. Additionally, we should migrate our software to ROS2. Furthermore, we need to measure the normal vector of the floor in real-time for the motion model, and test this on slopes or uneven terrain. We also want to experiment with more advanced Kalman-filter designs than just the linear version, such as the extended or unscented Kalman-filter.

CRediT authorship contribution statement

Fabian Arzberger: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Tim Schubert:** Software, Investigation, Data curation. **Fabian Wiecha:** Visualization, Methodology, Formal analysis, Data curation. **Jasper Zevering:** Software. **Julian Rothe:** Validation, Methodology. **Dorit Borrmann:** Validation, Supervision, Investigation, Conceptualization. **Sergio Montenegro:** Resources, Methodology. **Andreas Nüchter:** Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andreas Nuechter reports financial support was provided by European Space Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

A Github link is present in the paper where any software and datasets from the paper are available.

References

[1] F. Arzberger, F. Wiecha, J. Zevering, J. Rothe, D. Borrmann, S. Montenegro, A. Nüchter, Delta filter - robust visual-inertial pose estimation in real-time: A multi-trajectory filter on a spherical mobile mapping system, in: 2023 European Conference on Mobile Robots, ECMR, 2023, pp. 1–8, http://dx.doi.org/10.1109/ECMR59166.2023.10256359.

[2] R. Armour, K. Paskins, A. Bowyer, J. Vincent, W. Megill, Jumping robots: a biomimetic solution to locomotion across rough terrain, Bioinspir. Biomim. 2 (3) (2007) S65.

[3] K.W. Wait, P.J. Jackson, L.S. Smoot, Self locomotion of a spherical rolling robot using a novel deformable pneumatic method, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 3757–3762.

[4] R. Mukherjee, Spherical mobile robot, 2001, URL https://patents.google.com/patent/US6289263B1/en. US Patent 6, 289, 263.

[5] R. Chase, A. Pandya, A review of active mechanical driving principles of spherical robots, Robotics 1 (1) (2012) 3–23.

[6] D. Liu, H. Sun, Q. Jia, L. Wang, Motion control of a spherical mobile robot by feedback linearization, in: 2008 7th World Congress on Intelligent Control and Automation, 2008, pp. 965–970, http://dx.doi.org/10.1109/WCICA.2008.4593051.

[7] J. Zevering, D. Borrmann, A. Bredenbeck, A. Nuechter, Dynamics of spherical telescopic linear driven rotation robots, arXiv e-prints (2024) http://dx.doi.org/10.48550/arXiv.2404.09230, arXiv:2404.09230.
[8] D. Borrmann, S. Jörissen, A. Nüchter, RADLER – A RADial LasER scanning device, in: Proceedings of the International Symposium on Experimental Research, Buenos Aires, Argentina, 2020, pp. 655–664, http://dx.doi.org/10.1007/978-3-030-33950-0_56.
[9] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, A. Nüchter, Luna-a laser-mapping unidirectional navigation actuator, in: Experimental Robotics: The 17th International Symposium, Springer, 2021, pp. 85–94.
[10] A.P. Rossi, F. Maurelli, V. Unnithan, H. Dreger, K. Mathewos, N. Pradhan, D.-A. Corbeanu, R. Pozzobon, M. Massironi, S. Ferrari, et al., DAEDALUS-descent and exploration in deep autonomy of lava underground structures, 2021.
[11] J. Zevering, D. Borrmann, A. Bredenbeck, A. Nüchter, The concept of rod-driven locomotion for spherical lunar exploration robots, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2022, pp. 5656–5663, http://dx.doi.org/10.1109/IROS47612.2022.9981887.
[12] M. Bujňák, R. Pirník, K. Rástočný, A. Janota, D. Nemec, P. Kuchár, T. Tichý, Z. Łukasik, Spherical robots for special purposes: A review on current possibilities, Sensors 22 (4) (2022) http://dx.doi.org/10.3390/s22041413.
[13] F. Arzberger, A. Bredenbeck, J. Zevering, D. Borrmann, A. Nüchter, Towards spherical robots for mobile mapping in human made environments, ISPRS Open J. Photogr. Remote Sens. 1 (2021) 100004, http://dx.doi.org/10.1016/j.ophoto.2021.100004.
[14] F. Arzberger, J. Zevering, A. Bredenbeck, D. Borrmann, A. Nüchter, Mobile 3D scanning and mapping for freely rotating and vertically descended lidar, in: 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR, 2022, pp. 122–129, http://dx.doi.org/10.1109/SSRR56537.2022.10018586.
[15] J.-r. Xue, D. Wang, S.-y. Du, D.-x. Cui, Y. Huang, N.-n. Zheng, A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars, Front. Inf. Technol. Electron. Eng. 18 (1) (2017) 122–138.
[16] C. Merfels, C. Stachniss, Sensor fusion for self-localisation of automated vehicles, PFG–J. Photogr. Remote Sens. Geoinf. Sci. 85 (2017) 113–126.
[17] G. Abdi, F. Samadzadegan, F. Kurz, Pose estimation of unmanned aerial vehicles based on a vision-aided multi-sensor fusion, in: XXII ISPRS Congress, Technical Commission I, Vol. 41, 2016, pp. 193–199.
[18] H. Du, W. Wang, C. Xu, R. Xiao, C. Sun, Real-time onboard 3D state estimation of an unmanned aerial vehicle in multi-environments using multi-sensor data fusion, Sensors 20 (3) (2020) http://dx.doi.org/10.3390/s20030919.
[19] A. Soloviev, M.M. Miller, Navigation in difficult environments: Multi-sensor fusion techniques, in: V.L. Boginski, C.W. Commander, P.M. Pardalos, Y. Ye (Eds.), Sensors: Theory, Algorithms, and Applications, Springer New York, New York, NY, 2012, pp. 199–229, http://dx.doi.org/10.1007/978-0-387-88619-0_9.
[20] F. Santoso, M.A. Garratt, S.G. Anavatti, Visual–Inertial navigation systems for aerial robotics: Sensor fusion and technology, IEEE Trans. Autom. Sci. Eng. 14 (1) (2017) 260–275, http://dx.doi.org/10.1109/TASE.2016.2582752.
[21] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME–J. Basic Eng. 82 (Series D) (1960) 35–45.
[22] E. Wan, R. Van Der Merwe, The unscented Kalman filter for nonlinear estimation, in: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), 2000, pp. 153–158, http://dx.doi.org/10.1109/ASSPCC.2000.882463.
[23] F. Daum, Nonlinear filters: beyond the Kalman filter, IEEE Aerosp. Electron. Syst. Mag. 20 (8) (2005) 57–69, http://dx.doi.org/10.1109/MAES.2005.1499276.
[24] A.I. Mourikis, S.I. Roumeliotis, A multi-state constraint Kalman filter for vision-aided inertial navigation, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 3565–3572, http://dx.doi.org/10.1109/ROBOT.2007.364024.
[25] A. Sakai, Y. Tamura, Y. Kuroda, An efficient solution to 6DOF localization using unscented Kalman filter for planetary rovers, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4154–4159, http://dx.doi.org/10.1109/IROS.2009.5354677.
[26] G. Ligorio, A.M. Sabatini, Extended Kalman filter-based methods for pose estimation using visual, inertial and magnetic sensors: Comparative analysis and performance evaluation, Sensors 13 (2) (2013) 1919–1941, http://dx.doi.org/10.3390/s130201919.
[27] J. Kelly, G.S. Sukhatme, Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration, Int. J. Robot. Res. 30 (1) (2011) 56–79, http://dx.doi.org/10.1177/0278364910382802.
[28] G. Huang, K. Eckenhoff, J. Leonard, Optimal-state-constraint EKF for visual-inertial navigation, in: A. Bicchi, W. Burgard (Eds.), Robotics Research: Volume 1, Springer International Publishing, Cham, 2018, pp. 125–139, http://dx.doi.org/10.1007/978-3-319-51532-8_8.
[29] J. Liao, X. Li, X. Wang, S. Li, H. Wang, Enhancing navigation performance through visual-inertial odometry in GNSS-degraded environment, Gps Solut. 25 (2021) 1–18, http://dx.doi.org/10.1007/s10291-020-01056-0.
[30] N. Abdelkrim, N. Aouf, A. Tsourdos, B. White, Robust nonlinear filtering for INS/GPS UAV localization, in: 2008 16th Mediterranean Conference on Control and Automation, 2008, pp. 695–702, http://dx.doi.org/10.1109/MED.2008.4602149.
[31] M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2) (2002) 174–188, http://dx.doi.org/10.1109/78.978374.
[32] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, P.-J. Nordlund, Particle filters for positioning, navigation, and tracking, IEEE Trans. Signal Process. 50 (2) (2002) 425–437, http://dx.doi.org/10.1109/78.978396.
[33] A.J. Haug, Bayesian Estimation and Tracking: a Practical Guide, John Wiley & Sons, 2012, http://dx.doi.org/10.1002/9781118287798.
[34] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, M. Chli, GOMSF: Graph-optimization based multi-sensor fusion for robust UAV pose estimation, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 1421–1428, http://dx.doi.org/10.1109/ICRA.2018.8460193.
[35] T.-M. Nguyen, M. Cao, S. Yuan, Y. Lyu, T.H. Nguyen, L. Xie, VIRAL-fusion: A visual-inertial-ranging-lidar sensor fusion approach, IEEE Trans. Robot. 38 (2) (2022) 958–977, http://dx.doi.org/10.1109/TRO.2021.3094157.
[36] K. Kim, H.-T. Choi, C.-M. Lee, Underwater precise navigation using multiple sensor fusion, in: 2013 IEEE International Underwater Technology Symposium, UT, 2013, pp. 1–4, http://dx.doi.org/10.1109/UT.2013.6519855.
[37] W. Fang, L. Zheng, X. Wu, Multi-sensor based real-time 6-dof pose tracking for wearable augmented reality, Comput. Ind. 92–93 (2017) 91–103, http://dx.doi.org/10.1016/j.compind.2017.06.002.
[38] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, A. Nüchter, IMU-based pose-estimation for spherical robots with limited resources, in: 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI, IEEE, 2021, pp. 1–8.
[39] S. Madgwick, et al., An Efficient Orientation Filter for Inertial and Inertial/magnetic Sensor Arrays, Vol. 25 (2010) 113–118.
[40] H.G. Min, E.T. Jeung, Complementary Filter Design for Angle Estimation Using Mems Accelerometer and Gyroscope, Department of Control and Instrumentation, Changwon National University, Changwon, Korea, 2015, pp. 641–773.
[41] O.K. Vanjpe, M. Narasimhappa, A.D. Mahindrakar, Global Attitude Estimation and Dead Reckoning of a Mobile Spherical Robot Using Extended Kalman Filter, in: Proceedings of the 2019 4th International Conference on Advances in Robotics, AIR '19, Association for Computing Machinery, New York, NY, USA, 2020, http://dx.doi.org/10.1145/3352593.3352623.
[42] J. Borenstein, L. Feng, Gyrodometry: a new method for combining data from gyros and odometry in mobile robots, in: Proceedings of IEEE International Conference on Robotics and Automation, Vol. 1, 1996, pp. 423–428 vol.1, http://dx.doi.org/10.1109/ROBOT.1996.503813.
[43] F.L. Markley, Y. Cheng, J.L. Crassidis, Y. Oshman, Averaging quaternions, J. Guid. Control Dyn. 30 (4) (2007) 1193–1197, http://dx.doi.org/10.2514/1.28949.
[44] Phidgets, PhidgetSpatial precision 3/3/3 high resolution, 2023, https://phidgets.com/?&prodid=32#Tab_Specifications.
[45] Intel, Tracking camera T265 / T261 datasheet, 2019, https://dev.intelrealsense.com/docs/datasheets.
[46] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, A. Nüchter, IMU-based pose-estimation for spherical robots with limited resources, in: 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI, 2021, pp. 1–8, http://dx.doi.org/10.1109/MFI52462.2021.9591183.
[47] T. Qin, J. Pan, S. Cao, S. Shen, VINS-Fusion: An optimization-based multi-sensor state estimator, 2019, https://github.com/HKUST-Aerial-Robotics/VINS-Fusion.
[48] T. Qin, J. Pan, S. Cao, S. Shen, A general optimization-based framework for local odometry estimation with multiple sensors, 2019, https://arxiv.org/abs/1901.03638.
[49] T. Qin, P. Li, S. Shen, VINS-mono: A robust and versatile monocular visual-inertial state estimator, IEEE Trans. Robot. 34 (4) (2018) 1004–1020.
[50] F. Arzberger, 6-DoF Delta pose filter for sensor fusion, 2023, https://github.com/fallow24/delta_pose_filter.
[51] S. Umeyama, Least-squares estimation of transformation parameters between two point patterns, IEEE Trans. Pattern Anal. Mach. Intell. 13 (4) (1991) 376–380, http://dx.doi.org/10.1109/34.88573.
[52] M. Grupp, evo: Python package for the evaluation of odometry and SLAM, 2017, https://github.com/MichaelGrupp/evo.
[53] A. Nüchter, K. Lingemann, 3DTK—The 3D Toolkit. 2011, 2011, https://slam6d.sourceforge.io/index.html.
[54] A. Savitzky, M.J. Golay, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. 36 (8) (1964) 1627–1639, http://dx.doi.org/10.1021/ac60214a047.

Fabian Arzberger is a Ph.D. student at the University of Würzburg. He received his masters degree in the Elite Network Bavarian (ENB) program "Satellite Technologies", as well as a bachelors degree in "Aerospace Computer Science".

He graduated in 2021 at the chair of robotics and telematics, supervised by Prof. Andreas Nüchter, and stayed at the lab since then. His research focuses around 6-DoF pose estimation and LiDAR-based SLAM on spherical exploration robots.