Delta filter - robust visual-inertial pose estimation in real-time: A multi-trajectory filter on a spherical mobile mapping system*

Fabian Arzberger¹, Fabian Wiecha¹, Jasper Zevering¹, Julian Rothe², Dorit Borrmann³, Sergio Montenegro², and Andreas Nüchter¹

Abstract—Precise, real-time, and onboard 3D localization is mandatory for many robotic systems nowadays. Numerous ways exist to achieve this: Many state-of-the-art mobile mapping systems accomplish reliable and robust pose estimation utilizing combinations of inertial measurement units (IMUs), global navigation satellite systems (GNSS), visual-inertial- or LiDAR-inertial odometry (VIO/LIO). However, on a spherical mobile mapping system the underlying inherent rolling motion introduces high angular velocities, thus the quality of pose estimates, images, and laser-scans, degrade. In this work we propose a pose filter design that is able to do real-time sensor fusion between two unreliable trajectories - which are sampled asynchronously - into one, more reliable trajectory. It is a simple yet effective filter design that does not require the user to estimate the uncertainty of the sensors. The approach is not limited to spherical robots and theoretically is also suitable for sensor fusion of an arbitrary number of estimators. Furthermore, the filter does not refine its trajectory with information from future measurements and is able to run at 125 Hz on a Raspberry Pi 4. This work compares this filter against two pose estimation methods on our spherical system: (1) An approach that is based solely on IMU measurements and a motion model, and (2) stereo-VIO with an Intel® RealSense™ tracking camera. The proposed “Delta” filter takes as input (1), (2), and a motion model. Our implementation gets rid of the drift in (1) and (2), estimates the scale of the trajectory, and deals with slow and fast motion as well as driving curves. To quantify our results, we evaluate the trajectories against ground truth pose measurement using an OptiTrack™ motion capturing system. Furthermore, as our spherical system is equipped with a laser-scanner, we evaluate the resulting point clouds against ground truth maps available from a Riegli VZ-400 terrestrial laser scanner (TLS). Our source code can be found on github¹.

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. [1]–[6]. Using spherical robots for mobile mapping (cf. Figure 1) is a rather novel field. To the best of our knowledge, Borrmann et al. [7] first used a 2D laserscanner 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 [8] we used the same laserscanner 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 [9], [10]. 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. 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 errors in the resulting environment grow linearly, whereas for rotation these errors grow exponentially with increasing distance. While working with spherical robots, non-centered rotation is the main movement of the

Fig. 1: (Left) Spherical mobile mapping system equipped with Phidget IMUs, an Intel T265 stereo-tracking camera, Livox Mid-100 LiDAR, and optitrack IR-reflectors. (Right) Resulting point cloud when applying the estimated trajectory of the proposed Delta-filter, which uses information from the IMUs and internal camera, but not the LiDAR. The optitrack trajectories and resulting point clouds are used for evaluation. A video that demonstrates our implementation is available at https://youtu.be/2yu1RHtlesc
internal sensors, which proposes a huge challenge to state of the art SLAM algorithms. In previous work, we proposed initial offline-SLAM solutions for simplified sub-problems, i.e., rotation while descending [11], and rolling on flat surfaces [12]. However, in this work we address only the localization of the system and do not perform offline-SLAM, by introducing a pose estimation filter. Our implementation fuses information from three IMUs and a stereo-tracking camera onboard in real-time. The contributions of this work are as follows:

- A robust yet simple 6-DoF multi-trajectory filter, designed for but not limited to visual-inertial sensor fusion.
- An evaluation of our spherical mobile mapping systems accuracy based not only on ground truth point-clouds, but also on ground truth trajectories, which is stated as an open problem in [12].

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.

II. STATE-OF-THE-ART

Many onboard multi-sensor pose estimation approaches exist in the community. The majority of which being implemented and developed towards autonomous driving cars [13], [14], and unmanned aerial vehicles (UAV) [15], [16]. Soloviev et al. [17] 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. Santosoto et al. [18] categorize popular filters in the robotics community: (1) The Kalman Filter (KF) [19] 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 serval times, leading to variants such as the Unscented Kalman Filter (UKF) [20], Extended Kalman Filter (EKF) [21], or Multistate Constrained Kalman Filter (MSCKF) [22], 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 [16], [23]–[27]. (2) The $H_\infty$ 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 $H_\infty$ filter minimizes the worst-case estimation error, which leads to better performance if modelling uncertainties are present [28]. (3) Particle filters (PF), or Monte-Carlo Methods, are known for being applied in many stochastic estimation problems [29]. By now, it is well-known that PF outperforms KF in nonlinear systems underlying non-Gaussian noise [30]. 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 [31]. 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 [32] and VIRAL-Fusion [33], where the system states are represented and optimized in a pose-graph. The abovementioned 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. [34] 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. [35] 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 [36]. This approach [36] 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 [37] and Complementary filter [38]. 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 [36] is being utilized as input for the filter proposed in this paper. Lastly, we want to mention another filter that is much simpler than any of the approaches stated above, yet surprisingly effective: Gyrodometry [39]. 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. The proposed Delta-filter in this paper 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.

III. SENSOR FUSION WITH 6-DOF DELTA-FILTER

In this section we propose a new pose filter design: the “Delta” filter. Its purpose is to receive 6-DoF trajectory estimates from multiple sources, which are known to be
In our implementation we filter only two trajectory estimates with a given motion model, yet the Delta-filter is theoretically real-time. However, similar to a Kalman filter, the Delta-filter information from future measurements and is computed in time. Thus, the Delta-filter computes an estimate at the timestamps \( t \) and \( t_\tau \), by interpolating between two measurements at given timestamps \( t_\tau \), as shown in Figure 2. Note that rotation and translation parts by assuming that the measured and interpolated orientations are sufficiently reliable estimates, i.e., they dont drift or jump during a short time period. This assumption is valid for most inertial- and visual-tracking systems. To obtain the estimated filtered rotation delta \( \Delta R_e \), we compute

\[
\Delta R_e = \text{Slerp}\left( \Delta q_0, \Delta q_k, \frac{1}{2} \right)
\]

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

\[
\Delta X = \left[ R^{-1}(t_2) \cdot R(t_1), t(t_2) - t(t_1) \right]^T
\]

and estimate a new delta that makes more sense. 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

\[
\Delta X_m = [\Delta R_m, \Delta t_m]^T
\]

where \( f \) denotes the motion model that estimates the true motion given the measured and interpolated deltas, \( \Delta X_0 \) and \( \Delta X_k \). In a later section we will give an example for the motion model \( f \) when implementing the filter on a spherical robot.

1) Measurement, interpolation, and model: The measurement, interpolation, and model deltas \( \Delta X_0, \Delta X_k, \) and \( \Delta X_m \) respectively, are all considered unreliable. They are used to estimate the filtered pose \( \hat{X}_c(t_j) \) by iteratively applying an estimated filtered delta \( \Delta X_e \) that happened between \( t_{j-1} \) and \( t_j \):

\[
\hat{X}_c(t_j) = \Delta X_c \cdot \hat{X}_c(t_{j-1})
\]

\[
= [\Delta R_c \cdot R_c(t_{j-1}), \Delta t_c + t_c(t_{j-1})]^T
\]

We separate the rotation and translation parts by assuming that the measured and interpolated orientations are sufficiently reliable estimates, i.e., they dont drift or jump during a short time period. This assumption is valid for most inertial- and visual-tracking systems. To obtain the estimated filtered rotation delta \( \Delta R_e \), we compute

\[
\Delta R_e = \text{Slerp}\left( \Delta q_0, \Delta q_k, \frac{1}{2} \right)
\]

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

\[
\Delta X = \left[ R^{-1}(t_2) \cdot R(t_1), t(t_2) - t(t_1) \right]^T
\]

and estimate a new delta that makes more sense. 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

\[
\Delta X_m = [\Delta R_m, \Delta t_m]^T
\]

where \( f \) denotes the motion model that estimates the true motion given the measured and interpolated deltas, \( \Delta X_0 \) and \( \Delta X_k \). In a later section we will give an example for the motion model \( f \) when implementing the filter on a spherical robot.

1) Measurement, interpolation, and model: The measurement, interpolation, and model deltas \( \Delta X_0, \Delta X_k, \) and \( \Delta X_m \) respectively, are all considered unreliable. They are used to estimate the filtered pose \( \hat{X}_c(t_j) \) by iteratively applying an estimated filtered delta \( \Delta X_e \) that happened between \( t_{j-1} \) and \( t_j \):

\[
\hat{X}_c(t_j) = \Delta X_c \cdot \hat{X}_c(t_{j-1})
\]

\[
= [\Delta R_c \cdot R_c(t_{j-1}), \Delta t_c + t_c(t_{j-1})]^T
\]

We separate the rotation and translation parts by assuming that the measured and interpolated orientations are sufficiently reliable estimates, i.e., they dont drift or jump during a short time period. This assumption is valid for most inertial- and visual-tracking systems. To obtain the estimated filtered rotation delta \( \Delta R_e \), we compute

\[
\Delta R_e = \text{Slerp}\left( \Delta q_0, \Delta q_k, \frac{1}{2} \right)
\]

Note that for more than two estimators, the Slerp in Equation (9) must be replaced with a different quaternion average, e.g. [40]. Furthermore, we assume that the estimated translation deltas \( \Delta t_0, \Delta t_k, \) and \( \Delta t_m \) are not sufficiently reliable to just average them, as inertial-tracking tends to drift and visual-tracking tends to jump.

2) Probabilistic weighted geometric mean: There fore, we use a probabilistic approach that averages the translation direction and then scales it.

\[
\Delta t_e = \frac{d}{\sum_i \Delta t_i} \sum_i \Delta t_i
\]

where \( \Delta t_i \) refers to the measurement, interpolation, and model deltas. An estimate of the true scale of the translated distance \( d \) is given by a probabilistically weighted geometric mean:

\[
d = \left( \prod_i |\Delta t_i|^{\omega_i} \right) \left( \sum_i \omega_i \right)^{-1}
\]

We calculate weights \( \omega_i \) for each delta that correspond to the similarity of the deltas to their geometric mean, thus outliers

\[
\begin{align*}
\text{Interpolation } X_k \quad & \quad \hat{X}_c(t_\tau) \quad X_k(t_{\tau+1}) \\
\text{Measurement } X_0 \quad & \quad X_0(t_{\tau+1}) \quad X_0(t_\tau)
\end{align*}
\]

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 \( X_k \) get interpolated at measurement time \( t_\tau \).

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 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.

A. Proposed filter design

Suppose we have multiple 6-DoF pose estimators \( X = [R, t]^T \in SE(3) \), where \( R \) is a \( 3 \times 3 \) rotation matrix and \( t \) is a vector in \( \mathbb{R}^3 \). The pose of the \( k \)-th estimator at time \( t \) is denoted by \( X_k(t) = [R_k(t), t_k(t)]^T : \mathbb{R} \rightarrow SE(3) \). 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 \( X_0 \), yielding a query time \( t_\tau \). We call the resulting pose \( \hat{X}_c(t_\tau) \) the “measurement”. All other estimators \( X_k \) are queried at time \( t_\tau \), by interpolating between two measurements at given timestamps \( t_k \), as shown in Figure 2. Note that rotation matrices and unit quaternions are isomorphic, thus we use \( q_k(t) \) and \( \hat{R}_k(t) \) interchangeably as they represent the same elements in \( \text{SO}(3) \).

The interpolation is constructed using quaternion slerp and linear vector interpolation as described by Equations (1) - (5):

\[
X_k(t_\tau) = [R_k(t_\tau), t_k(t_\tau)]^T,
\]

\[
\hat{t} = \frac{t_\tau - t_{\tau+1}}{t_{\tau-1} - t_{\tau+1}} \in [0; 1],
\]

\[
\Omega = \cos^{-1}(q_k(t_{\tau-1}) \cdot q_k(t_{\tau+1})),
\]

\[
\hat{R}_k(t_\tau) = \text{Slerp}\left( q_k(t_{\tau-1}), q_k(t_{\tau+1}), \hat{t} \right)
\]

\[
= \frac{\sin((1 - \hat{t})\Omega)}{\sin(\Omega)} \cdot q_k(t_{\tau-1}) + \frac{\sin(\hat{t}\Omega)}{\sin(\Omega)} \cdot q_k(t_{\tau+1}),
\]

\[
t_k(t_\tau) = (1 - \hat{t}) \cdot t_k(t_{\tau-1}) + \hat{t} \cdot t_k(t_{\tau+1})
\]
get a damped weighting while similar values get a higher weighting:

\[
| \hat{t} | = \left( \prod_{i=1}^{n} | \Delta t_i | \right)^{n^{-1}},
\]

(12)

\[
s = \sqrt{\frac{1}{n-1} \sum_{i=0}^{n} \left( | \Delta t_i | - | \hat{t} | \right)^2},
\]

(13)

\[
w_i = 1 - s^{-1} \cdot ( | \Delta t_i | - | \hat{t} | )
\]

(14)

B. Implementation on a spherical system

The implementation on our spherical robot uses two estimators, the IMU operating at 125 Hz defines the measurement \( X_0 \), and the camera operating at 200 Hz defines the interpolation \( X_k \). 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:

\[
f(\Delta X_0, \Delta X_k) = \left[ \Delta R_e, r \cdot \angle (\Delta R_e) \cdot \left( \frac{\Delta t_0 + \Delta t_k}{| \Delta t_0 + \Delta t_k |} \right) \right]^\top,
\]

(15)

where \( \angle (\cdot) \) denotes the angle around the axis described by the rotation matrix. Note that we just defined the model rotation \( \Delta R_m \) from Equation (9) to be equal to \( \Delta R_e \) from Equation (9), as the orientation estimation is considered sufficiently reliable.

The simplicity of the filter design allows for the introduction of simple but effective design choices. As an example we notice that our IMUs tends to drift without the use of a magnetometer, especially in the yaw-axis, whereas the tracking camera does not. 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 Equation (9). 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.

IV. EXPERIMENTS AND EVALUATION

Qualitative results are presented in Figure 3 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 unfavourable type of sensor motion. Our proposed Delta-filter (c) combines both trajectories in real-time at 125 Hz on a Raspberry Pi 4, gets rid of the drift and jumps, and estimates the scale of the trajectory better. The following sections quantify the results using ground truth trajectories and maps.

A. 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 \( X_{\text{ref}} = [R_{\text{ref}}, t_{\text{ref}}] \), and the other estimated trajectories \( X_{\text{est}} = [R_{\text{est}}, t_{\text{est}}] \). 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 [42]. Note that we only use the estimated rotation of the Umeyama method, since we already aligned the origins. From this point, we use Grupps [43] software for trajectory evaluation. The resulting point clouds are aligned to ground truth using the well-known Iterative Closest Points (ICP) algorithm. We use 3DJK [44] for the processing of the point clouds.

1) Absolute position error: The absolute position error (APE) represents the error of the translation estimation and is given by

\[
\text{APE}_t = | t_{\text{est},i} - t_{\text{ref},i} | \ [m] .
\]

(16)

2) Relative pose error: The relative pose error (RPE) represents the error of the orientation estimation and is given by

\[
\text{RPE}_e = \left| \angle \left( R_{\text{ref},i}^{-1} R_{\text{est},i-1}^{-1} \right) R_{\text{est},i}^{-1} R_{\text{est},i-1} \right| \ [\text{deg}] .
\]

(17)

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 \( m_i, d_i \in \mathbb{R}^3 \) respectively. Then, the root mean squared error is given by

\[
\text{RMSE} = \frac{1}{N} \sum_{i=0}^{N} | m_i - d_i |^2
\]

(18)

B. Experiments

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 Equations (17) and (16). 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 mm to provide accurate ground truth point clouds. As our system is equipped with a laser scanner (cf. Figure 3), we compare the resulting point cloud to the ground truth map using Equation (18). Both setups are shown in Figure 5.

1) Fast motion: In this experiment, the sphere traversed a distance of approx. 4 m in about 10 s. Figure 6 shows the APE (15) 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 shows a considerable increase of error due to the accumulated drift. The error of the proposed Delta-filter are orders of magnitude smaller compared to the IMUs and T265. Figure 7 shows the comparison of RPE (17) over time. Note that the Savgol-filter [45] is applied to the error signals. This is because 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
TABLE I: 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.

<table>
<thead>
<tr>
<th>Estimator</th>
<th>RMSE [m]</th>
<th>Mean [m]</th>
<th>Std. [m]</th>
<th>Max. [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slow Fast</td>
<td>Slow Fast</td>
<td>Slow Fast</td>
<td>Slow Fast</td>
</tr>
<tr>
<td>Dead-reckoning INS</td>
<td>1.713 1.736</td>
<td>1.447 1.291</td>
<td>0.917 1.160</td>
<td>2.882 3.001</td>
</tr>
<tr>
<td>Intel T265 Stereo-VIO</td>
<td>4.486 7.441</td>
<td>4.012 5.290</td>
<td>2.008 5.234</td>
<td>5.848 13.549</td>
</tr>
<tr>
<td>Proposed Delta-filter</td>
<td>0.114 0.248</td>
<td>0.103 0.193</td>
<td>0.049 0.165</td>
<td>0.189 0.428</td>
</tr>
</tbody>
</table>

TABLE II: 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slow Fast</td>
<td>Slow Fast</td>
<td>Slow Fast</td>
<td>Slow Fast</td>
</tr>
<tr>
<td>Dead-reckoning INS</td>
<td>1.389 4.318</td>
<td>1.281 3.270</td>
<td>0.537 2.819</td>
<td>2.653 8.883</td>
</tr>
<tr>
<td>Intel T265 Stereo-VIO</td>
<td>1.374 4.213</td>
<td>1.264 3.190</td>
<td>0.541 2.753</td>
<td>2.701 8.852</td>
</tr>
<tr>
<td>Proposed Delta-filter</td>
<td>1.384 4.305</td>
<td>1.273 3.248</td>
<td>0.543 2.825</td>
<td>2.752 9.199</td>
</tr>
</tbody>
</table>

estimators do not differ particularly from each other, which is also evident from the error metrics in Table I. 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 Equation (9).

2) Slow motion: In this experiment, the sphere traversed a distance of approx. 4 m in about 45 s. Figure 8 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 I confirms this observation. Figure 9 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.

3) Curves: Figure 10 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
C. Discussion

The evaluation shows that the Delta-filter significantly improves the pose estimation accuracy, reduces drift, and eliminates jumps. However, despite reducing the drift, all experiments show that the filter still suffers from global drift regarding translation. Furthermore, in the resulting point clouds, the walls appear to be thicker than in the ground truth point cloud, which comes down to two factors: First, the Livox Mid-100 used in the experiment has 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.

V. 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. We therefore proposed the simple yet effective Delta-filter, which is able to do real-time sensor fusion of an INS- with a VIO-based solution. The filter needs a motion model defined by the user, greatly decreases the INS drift, and gets rid of the jumps caused by the VIO. We showed that the filter is reliable in slow and fast motion, as well as driving curves. Furthermore, we estimated the mapping accuracy of the spherical mobile mapping system to be 18.6 cm without the use of offline-SLAM, which is considered to be an improvement to our previous work. Having such a trajectory estimate brings real-time, highly precise laser-based SLAM for spherical robots closer to reality in the near future. However, needlessly to say, a lot of work remains to be done. In the future, we need to address a proper extrinsic calibration between all sensors to further increase the accuracy. We will also incorporate the LiDAR measurements into the localization by building a real-time onboard laser-based SLAM algorithm designed for spherical systems. This will also include the extension of the motion model using environment data to account for slopes, uneven terrain, or free falling for a short period of time.

REFERENCES


[38] Min, Hyung Gi and Jeung, Eun Tae, “Complementary filter design for angle estimation using mems accelerometer and gyroscope,” Department of Control and Instrumentation, Changwon National University, Changwon, Korea, pp. 641–773, 2015.


