Anderson Acceleration for on-Manifold Iterated Error State Kalman Filters

Xiang Gao[1], Tao Xiao[1], Chunge Bai[1,2], Dezhao Zhang[1] and Fang Zhang[1]

Abstract— Iterated Extended Kalman Filter is a promising and widely-used estimator for real-time localization applications. It iterates the observation equation to find a better linearization point and, simultaneously, only maintains the state estimation in a single time to save the computation resources. Inspired by the recent development of the iterative closest point algorithm, this paper investigates an acceleration approach to the iterations in iterative error state Kalman filters (IESKFs). We show that the IESKF can be seen as a fixed point problem, and the Anderson acceleration (AA) can be elegantly applied to the iterations of IESKF since the error state naturally lies in the tangent space and does not require additional transforms. However, the tangent space of the current estimation may change during the iterations, so we should switch the tangent space to the starting point to perform the Anderson acceleration. We propose the AA-IEKF and apply it to the lidar-inertial odometry (LIO) systems to estimate the ego motion of a lidar. The experiments show that the Anderson acceleration can efficiently reduce the number of iterations in ESKF and achieve a lower computational cost.

I. Introduction

Extended Kalman filter (EKF) is one of the most important sequential state estimators in simultaneous localization and mapping (SLAM) research from early Lidar/vision-based SLAM systems [1], [2] to recent Lidar-inertial SLAM systems like the FastLIO series [3]–[5]. EKF linearizes the motion and observation equations and finds an optimal solution between the predicted prior and the observation data [6], [7]. Many previous studies have thoroughly discussed the advantages and disadvantages of EKF [8], [9]. For example, EKF is real-time and fast, but its Markov property makes it hard to tackle long-time associations like large-scale loop closing [10]. If the landmarks are also considered unknown states to estimate, the dimension of EKF would be too large for real-time applications. The covariance matrix would then be dense, and the inversion of a dense matrix would be $O(3)$ in this case. From the optimization perspective, EKF can also be regarded as a single-time, non-iterating factor graph with three factors (the prior, the motion, and the observation) that immediately marginalizes the previous state [11].

This work was supported by Idriver+ Technologies Co. Ltd. in Beijing, China.

[1]Xiang Gao, Chunge Bai, Tao Xiao, Dezhao Zhang and Fang Zhang are with the Idriver+ Technologies Co. Ltd. in Beijing, China. Email: gao.xiang.thu@gmail.com

[2]Chunge Bai is with the Shenzhen International Graduate School, Tsinghua University, Shenzhen, China. Email: bcg20@mails.tsinghua.edu.cn

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

However, in many practical applications where the map is pre-built or considered fixed, EKF is still a favored choice if the landmarks are accurate enough. In that case, we only need to estimate the robot's/vehicle's pose and its uncertainty instead of all the landmarks. In Lidar odometry (LO) or Lidar-inertial odometry (LIO) algorithms, EKF is a prevalent backend estimator choice since the lidar point clouds are usually accurate enough to build a local map [12]. There are also many recent studies focusing on improving the performance of traditional EKF. For example, the error state Kalman filter (ESKF) is normally a better alternative to EKF since the error state is typically close to zero to make better linearization [13], [14]. The observation equation could be iterated instead of directly merged into the estimator [15]. Multiple states can also be integrated into a single EKF to make a longer estimation window to prevent the system from falling into an erroneous working point [16], [17]. As a result, IEKF/MSCKF acts as a compromise and practical choice between the traditional, single-time Kalman filter and the full-states offline optimization [18].

The EKF in LIO or VIO systems usually estimates a high-dimensional state rather than the simple Euler angles plus a translation vector in early research [19]. The state variable, in these cases, lies in a high-dimensional manifold $\mathcal{M}$, and the linearization is performed in the tangent space of the current estimation. Such on-manifold linearizations are widely used in modern filters and optimizers [20], [21]. If the EKF or EKSF in LIO is iterated, the observation model would be very similar to the iterative closes point (ICP) process, where we will alternatively find the closest neighbors and solve for an optimal solution [22], [23]. The original ICP aims to find the best pose $\mathbf{T} \in \mathrm{SE}(3)$, but ESKF will find the Maximum a Posteriori (MAP) solution between the prediction and the observation [24]. On the other hand, the ICP can also be regarded as a majorization-minimization algorithm, where the solution of the nearest neighbor (the correspondence estimation step) in each iteration is a surrogate function of the original function [25]. The Anderson acceleration [26] and other local quadratic approximates can be employed to improve its convergence speed into second-order rather than the linear speed in the original ICP [27], [28].

In this paper, we show that the IESKF could be converted to a fixed point problem, and we can integrate the Anderson acceleration into an IESKF to obtain better convergence speed. The recently-proposed AA-

ICP [29] or FRICP [25] either uses the Euler angles that may suffer from singularities or require the logarithm map to convert the on-manifold pose to its Lie algebra. The error state in the IEKF naturally lies in the tangent space that does not need extra conversion. However, when we perform iterations on the observation model, the tangent space of the current estimation will change with the iteration, which brings additional Jacobian matrices to switch those tangent spaces. To perform the AA of the error state, we should project the estimation into the tangent space of the start point in IEKF.

The contribution of this paper can be summaries as follows:

1) We give the formulation of applying AA in iterated error state Kalman filter (IESKF).
2) We tackle the problem of switching the tangent spaces during iteration, which is different in AA-ICP or similar methods.
3) We also utilize the AA-IEKF to build an LIO system and test its convergence speed and accuracy. We show that the AA-IEKF achieves better convergence speed and less computation cost, which is practically meaningful for future applications.

II. Related Work

The SLAM problem was formulated as a state estimation problem solved by various filters in early research, e.g., ESKF [14], particle filters (RBPF) [30], and many other variants [31]. Many research focus on the theoretic characteristics of EKF, such as observability and inner consistency [8], [32]. It was later shown that the sequential filter-like approaches are to some degree equal to the nonlinear optimization approaches, such as the bundle adjustment in VSLAM [18], which can be further interpreted as a graph optimization or factor graph [33]. However, for LIO or LO, the bundle adjustment-like approaches are normally more expensive in computation than the filter-like approaches since the projection of the landmark relationships will be explicitly modeled [34], [35]. Many prevalent Lidar SLAM systems like LOAM [12] and LeGO-LOAM [36] still treat landmarks as static point clouds and only estimate the lidar poses.

On the other hand, recent developments in point cloud registration show that the classical ICP can be accelerated using a quadratic approximation [37] or ℓ_p norm minimization [38]. Furthermore, ICP can also be treated as a fixed-point iteration process that can be accelerated by Anderson acceleration [25], [29], which was initially proposed by Anderson Donald for solving general nonlinear integral equations [26], [39]. The accelerated ICP shows a faster convergence speed in these studies, but the problem is usually formulated as Euler angles or Lie algebra that needs an additional logarithm map from the manifold. Inspired by these studies, we propose the on-manifold Anderson accelerated IEKF in this paper and adapt it into a lidar-inertial odometry system.

III. On-Manifold Error State Kalman Filter

A. Formulation

We first give the derivation of the ESKF here and then discuss the prediction and update process in detail. The state variable is defined on a high-dimensional manifold $\mathcal{M}$:

$$\mathbf{x} = [\mathbf{p}, \mathbf{R}, \mathbf{v}, \mathbf{b}_g, \mathbf{b}_a, \mathbf{g}]^T \in \mathcal{M}, \tag{1}$$

where $\mathbf{p}$ is the position, $\mathbf{R}$ is the rotation, $\mathbf{v}$ is the velocity, $\mathbf{b}_g$ and $\mathbf{b}_a$ are IMU biases, and $\mathbf{g}$ is the gravity. All the variables are defined as vectors/matrices transforming the body frame to the world frame, e.g., $\mathbf{R}_{WB}$, so the subscripts are omitted. The state variable is a Cartesian product of $\mathcal{M} = \mathbb{R}^3 \times \mathrm{SO}(3) \times \mathbb{R}^9 \times \mathrm{S}(2)$. The order of the state variables is arbitrary, but the form of the Jacobian matrices will depend on the order of the states, so we assume the states are stored in a fixed order.

In ESKF, the true state variable $\mathbf{x}_t$ can be divided into the nominal state and the error state:

$$\mathbf{x}_t = \mathbf{x} \boxplus \delta\mathbf{x}, \tag{2}$$

where $\mathbf{x}_t$ is the true state, $\mathbf{x}$ is the nominal state, and $\delta\mathbf{x}$ is the error state. The $\boxplus$ operation is defined between the manifold and the tangent space: $\boxplus : \mathcal{M} \boxplus \mathbb{V} \to \mathcal{M}$ [40]. For variables that lie in the vector space, such as $\mathbf{p}$, it is the same as $\mathbf{p} + \delta\mathbf{p}$. For $\mathbf{R} \in \mathrm{SO}(3)$, it is defined with the exponential map:

$$\mathbf{R}_t = \mathbf{R}\mathrm{Exp}(\delta\mathbf{R}), \tag{3}$$

where $\delta\mathbf{R}$ is typically denoted as $\delta\boldsymbol{\theta} \in \mathbb{R}^3$. The exponential map of $\mathrm{SO}(3)$ is the same as Rogridues' formula. Let $\boldsymbol{\theta}$ be decomposed by its norm and unit-length direction $\boldsymbol{\theta} = \theta\mathbf{n}$, then the exponential map from $\mathfrak{so}(3)$ to $\mathrm{SO}(3)$ is:

$$\begin{aligned} \mathrm{Exp}(\boldsymbol{\theta}) &= \exp(\boldsymbol{\theta}^\wedge) \\ &= \cos\theta\mathbf{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta\mathbf{n}^\wedge, \end{aligned} \tag{4}$$

where $^\wedge$ is an operator that converts a vector to its skew-symmetric matrix.

For $\mathbf{x} \in \mathrm{S}(2)$, the $\delta\mathbf{x}$ is defined in $\mathbb{R}^2$, and the $\boxplus$ operator is defined as:

$$\mathbf{x} \boxplus \delta\mathbf{x} = \mathrm{Exp}(\mathbf{B}(\mathbf{x})\delta\mathbf{x})\mathbf{x}, \tag{5}$$

where $\mathbf{B}(\mathbf{x}) \in \mathbb{R}^{3\times 2}$ is given in the equation (39) in the appendix.

ESKF maintains the mean of the nominal state $\mathbf{x}$ and the covariance of the error state $\delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$. The details of the state dynamics are given in the appendix section (Eq. 32). When the IMU data $\mathbf{u}$ and lidar data $\mathbf{z}$ come, the inertial measurements will be integrated into the nominal state in the prediction stage, and the noise and observation will be added into the error state in the update stage. Finally, the error stated will be added to the predicted nominal state and used as the full state estimation. Readers can also refer to [13] for a more elaborate description of the ESKF pipeline.

B. IEKF Prediction

The Kalman filter is formed in two stages: prediction and update. The prediction is not iterated since we only have one group of IMU measurements. Assume we are moving from time k to $k+1$, the prediction can be written as:

$$\mathbf{x}_{\text{pred}}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)), \tag{6}$$

where $\mathbf{f}$ is the abstract motion equation and $\mathbf{u}$ is the abstract input. The noise items are tackled in the error state equation:

$$\delta\mathbf{x}_{\text{pred}}(k+1) = \mathbf{f}'(\delta\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)), \tag{7}$$

where $\mathbf{f}'$ is the nonlinear motion equation of the error state and $\mathbf{w}$ is the noise vector. Using the definition of (1), we can write out the exact form of the error state dynamics (see equation (32) of Appendix A). To compute the predicted covariance, we linearize the error state dynamics as:

$$\delta\mathbf{x}_{\text{pred}}(k+1) = \mathbf{F}(k)\delta\mathbf{x}(k) + \mathbf{w}'(k), \tag{8}$$

where $\mathbf{F}$ is the Jacobian matrix of $\mathbf{f}'$ and $\mathbf{w}' \sim \mathcal{N}(\mathbf{0}, \mathbf{W})$ is the linearized Gaussian noise item. The covariance matrix of the prediction will be:

$$\mathbf{P}_{\text{pred}}(k) = \mathbf{F}(k)\mathbf{P}(k)\mathbf{F}^T(k) + \mathbf{W}(k), \tag{9}$$

which is just a linear transform of state estimation at the last time. We will drop the k in the following text since the discussion is always limited in time k:

$$\mathbf{P}_{\text{pred}} = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{W}. \tag{10}$$

C. IEKF Update

The abstract observation equation on the nominal state can be written as:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v}, \tag{11}$$

where $\mathbf{h}$ is the abstract observation equation, $\mathbf{z}$ is the observed data, and $\mathbf{v}$ is the noise variable satisfying $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$.

The Jacobian matrix of the error state can be solved using the chain rule:

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \delta\mathbf{x}}, \tag{12}$$

where the latter item can be written out using the definition of the full state $\mathbf{x}$:

$$\frac{\partial \mathbf{x}}{\partial \delta\mathbf{x}} = \mathbf{I}, \text{for } \mathbf{x} \in \mathbb{R}^3, \tag{13}$$

$$\frac{\partial \mathbf{R}}{\partial \delta\boldsymbol{\theta}} = \frac{\partial \text{Log}(\mathbf{R}\text{Exp}(\delta\boldsymbol{\theta}))}{\partial \delta\boldsymbol{\theta}} = \mathbf{J}_r^{-1}(\mathbf{R}), \tag{14}$$

$$\frac{\partial \mathbf{x}}{\partial \delta\mathbf{x}} = \mathbf{M}(\mathbf{x} \boxplus \delta\mathbf{x}) \in \mathbb{R}^{3\times 2} \text{ for } \mathbf{x} \in \text{S}(2), \tag{15}$$

where $\mathbf{J}_r^{-1}(\mathbf{R}) \in \mathbb{R}^{3\times 3}$ is the inverse of the right Jacobian matrix in SO(3), which could be derived from the linear form of the Baker-Campbell-Hausdorff formula [7], and $\mathbf{M}$ is given in (40) in the appendix.

The update step can be iterated in different linearization points, so we will denote the state variable of the j-th iteration as $\mathbf{x}^j$, the incremental part as $\delta\mathbf{x}^j$, and the start point as $\mathbf{x}^0$. Note that since the linearization point will change in each iteration, we need to project the resulting Gaussian estimation in the tangent space from the starting point $\mathbf{x}^0$ to $\mathbf{x}^j$, which gives an extra Jacobian matrix $\mathbf{J}^j$:

$$\mathbf{J}^j = \frac{\partial \mathbf{x}^j \boxplus \delta\mathbf{x} \boxminus \mathbf{x}^j}{\partial \delta\mathbf{x}}. \tag{16}$$

In this way, we can simply convert the predicted covariance matrix $\mathbf{P}_{\text{pred}}$ into the current tangent space:

$$\mathbf{P}_c^j = \mathbf{J}^j \mathbf{P}_{\text{pred}} (\mathbf{J}^j)^T, \tag{17}$$

and also the error state:

$$\delta\mathbf{x}_c^j = \mathbf{J}^j(\mathbf{x}^j \boxminus \mathbf{x}^0), \tag{18}$$

where the subscript c denotes the variable in the current tangent space. The j-th update step can be summarized as:

$$\mathbf{K}^j = \mathbf{P}_c^j (\mathbf{H}^j)^T (\mathbf{H}^j \mathbf{P}_c^j (\mathbf{H}^j)^T + \mathbf{V})^{-1}, \tag{19}$$

$$\delta\mathbf{x}^{j+1} = \mathbf{K}^j(\mathbf{z} - \mathbf{h}(\mathbf{x}^j) + \mathbf{H}^j \delta\mathbf{x}_c^j) - \delta\mathbf{x}_c^j, \tag{20}$$

which is the same as the traditional Kalman filter, except the error state is used here instead of the nominal state.

Then the current estimation of the nominal state is updated using:

$$\mathbf{x}^{j+1} = \mathbf{x}^j \boxplus \delta\mathbf{x}^{j+1}, \tag{21}$$

and finally, if the filter converges, we will get the posterior mean and covariance estimation. The mean is just the $\mathbf{x}^{j+1}$, where j is the last iteration number. But for the covariance matrix, we need to project $\mathbf{P}^{j+1}$ into the tangent space of $\mathbf{x}^{j+1}$:

$$\mathbf{P}^{j+1} = (\mathbf{I} - \mathbf{K}^j \mathbf{H}^j) \mathbf{J}^j \mathbf{P}_{\text{pred}} (\mathbf{J}^j)^T, \tag{22}$$

$$\mathbf{P}_{\text{update}} = \mathbf{L}\mathbf{P}^{j+1}\mathbf{L}^T, \tag{23}$$

where

$$\mathbf{L} = \frac{\partial \mathbf{x}^{j+1} \boxplus \delta\mathbf{x} \boxminus \mathbf{x}^{j+1}}{\partial \delta\mathbf{x}}. \tag{24}$$

IV. Anderson Acceleration on IEKF

A. The AA Iteration

The fixed point problem is to find a variable $\mathbf{u}$ that satisfies the equation $\mathbf{u} = \mathbf{G}(\mathbf{u})$. Given the initial value $\mathbf{u}_0$ of $\mathbf{u}$, the solution of $\mathbf{u}$ can be accessed by iteratively solving $\mathbf{u}_1 = \mathbf{G}(\mathbf{u}_0), \ldots, \mathbf{u}_{n+1} = \mathbf{G}(\mathbf{u}_n)$. In IESKF, we take the error state $\delta\mathbf{x}$ as $\mathbf{u}$ since the updated nominal state estimation $\mathbf{x}^j$ converges to the true state. If IEKF converges, the incremental part $\mathbf{x}^j \boxminus \mathbf{x}^0$ will also converge

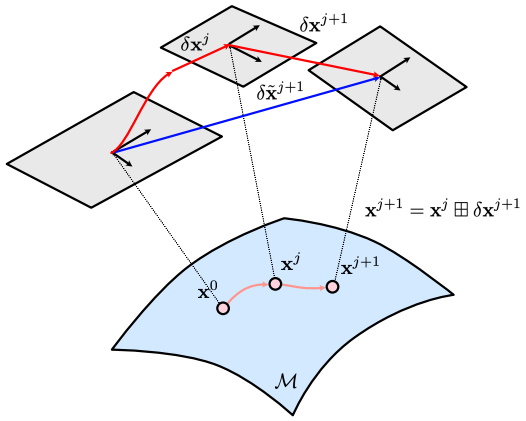

Fig. 1. The relationship of the IEKF update $\delta\mathbf{x}^{j+1}$ and the Anderson accelerated update $\delta\tilde{\mathbf{x}}^{j+1}$. Note that the AA update is always in the tangent space of $\mathbf{x}^0$.

in that case. So the j-th estimation of the error state can be regarded as the best estimation of:

$$(\delta\mathbf{x}^{j+1})^* = \arg\min_{\delta\mathbf{x}} \underbrace{\|\mathbf{z} - \mathbf{H}^j(\mathbf{x}^j \boxplus \delta\mathbf{x})\|^2_{\mathbf{V}}}_{\text{observation}} + \underbrace{\left\|\mathbf{x}^j \boxminus \mathbf{x}^0 + (\mathbf{J}^j)^{-1}\delta\mathbf{x}\right\|^2_{\mathbf{P}_{\text{pred}}}}_{\text{prior}}, \tag{25}$$

where $\|\cdot\|_{\mathbf{P}}$ means taking the Mahalanobis distance with the covariance matrix $\mathbf{P}$. The above least-square problem is solved by the Kalman gain and updating using equation (19).

AA aims at accelerating the solving stage of the fixed point problem. The basic idea is to use historical iterations to guess the current iteration. We can accelerate the convergence by combining the previous iterating directions by multiplying the weights θ_i. However, to perform the AA in IESKF, we should use a variable in the same tangent space that does not move during the iterations. Therefore, we use a slightly different $\delta\tilde{\mathbf{x}}^j$ that is always defined in the tangent space of the start point (see Fig. 1):

$$\delta\tilde{\mathbf{x}}^j = \mathbf{x}^j \boxplus \delta\mathbf{x}^j \boxminus \mathbf{x}^0. \tag{26}$$

With the definition of $\delta\tilde{\mathbf{x}}^j$, the update of the mean estimation will be:

$$\mathbf{x}^{j+1} = \mathbf{x}^0 \boxplus \delta\tilde{\mathbf{x}}^{j+1} \tag{27}$$

Unlike the formulation in [25] and [38], the error state $\delta\tilde{\mathbf{x}}^{j+1}$ naturally lies in the tangent space of $\mathbf{x}_{\text{pred}}$, which is a well-formed vector space. The general AA will calculate the j-th iteration using the weighted m previous iterates $\delta\tilde{\mathbf{x}}^{j-m}, \ldots, \delta\tilde{\mathbf{x}}^j$:

$$\delta\tilde{\mathbf{x}}^{j+1}_{\text{AA}} = \mathbf{G}(\delta\tilde{\mathbf{x}}^j_{\text{AA}}) - \sum_{i=1}^{m} \theta^*_i(\mathbf{G}(\delta\tilde{\mathbf{x}}^{j-i+1}_{\text{AA}} - \mathbf{G}(\delta\tilde{\mathbf{x}}^{j-i}_{\text{AA}}))), \tag{28}$$

where $\theta^*_1, \ldots, \theta^*_m$ are the paramters estimated by the following linear least-square equation:

$$(\theta^*_1, \ldots, \theta^*_m) = \arg\min \left\| \mathbf{G}_j - \sum_{i=1}^{m} \theta_i(\mathbf{G}_{j-i+1} - \mathbf{G}_{j-i}) \right\|^2, \tag{29}$$

where $\mathbf{G}_j$ is short for $\mathbf{G}(\delta\tilde{\mathbf{x}}^j_{\text{AA}})$.

In AA-IEKF, we will try the AA iteration first and use the point-to-plane residuals in the lidar observation equation as an indicator of accepting the AA iteration. If the AA iteration gives an estimation worse than the previous iteration, we will directly use the current $\delta\mathbf{x}^j$ as the best estimation rather than $\delta\tilde{\mathbf{x}}^j_{AA}$. The algorithm is summarized in Algorithm 1.

Algorithm 1 AA-IEKF

Input: mean and cov of last time: $\mathbf{x}(k-1), \mathbf{P}(k-1)$
Output: mean and cov of this time: $\mathbf{x}(k), \mathbf{P}(k)$

1: Prediction:
2: $\mathbf{P}_{\text{pred}} = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}$
3: $\mathbf{x}^0 = \mathbf{f}(\mathbf{x}, \mathbf{u}), \delta\mathbf{x}^0 = \mathbf{0}$
4: Iteratively update starts from $\mathbf{x}^0$
5: for j = 0, $\ldots, n$ do
6: Project $\delta\mathbf{x}^j$ into the tangent space of $\mathbf{x}^j$:
7: Compute $\mathbf{J}^j$ with (16)
8: Set $\delta\mathbf{x}^j_c = \mathbf{J}^j(\mathbf{x}^j \boxminus \mathbf{x}^0), \mathbf{P}^j_c = \mathbf{J}^j\mathbf{P}_{\text{pred}}(\mathbf{J}^j)^T$
9: Compute $\mathbf{Q}^j = (\mathbf{H}^j)^T\mathbf{V}^{-1}\mathbf{H}^j + (\mathbf{P}^j_c)^{-1}$
10: Compute $\mathbf{K}^j = (\mathbf{Q}^j)^{-1}(\mathbf{H}^j)^T\mathbf{R}^{-1}$
11: Set $\delta\mathbf{x}^{j+1} = \mathbf{K}^j(\mathbf{z} - \mathbf{H}^j\mathbf{x}^j) + (\mathbf{K}^j\mathbf{H}^j - \mathbf{I})\delta\mathbf{x}^j_c$
12: Set $\mathbf{x}^{j+1} = \mathbf{x}^j \boxplus \delta\mathbf{x}^{j+1}$
13: Set $\delta\tilde{\mathbf{x}}^{j+1} = \mathbf{x}^{j+1} \boxminus \mathbf{x}^0$
14: Compute $\delta\tilde{\mathbf{x}}^{j+1}_{\text{AA}}$ using (28).
15: Set $\mathbf{x}^{j+1} = \mathbf{x}^0 \boxplus \delta\tilde{\mathbf{x}}^{j+1}_{\text{AA}}$.
16: if algorithm converges then
17: Compute $\mathbf{P}^{j+1} = (\mathbf{I} - \mathbf{K}^j\mathbf{H}^j)\mathbf{J}^j\mathbf{P}_{\text{pred}}$
18: Update the final covariance.
19: else
20: return to 5.
21: end if
22: end for
23: return $\mathbf{x}^{j+1}, \mathbf{P}_{\text{update}}$

In practice, the prediction will be triggered by high-frequency IMU input, and the update step is triggered by lidar point cloud matching. The Jacobian matrices of the point-to-plane residuals are given in the appendix section. The convergence condition is set as $\forall i \in [1, d], |\delta\mathbf{x}^j_i| < 10^{-3}$, where d is the dimension of $\delta\mathbf{x}^j$.

B. First Estimate Jacobian (FEJ) and Inconsistency

There has been a well-known issue about EKF. The filter may converge to an over-confident solution if the Jacobian matrices are not evaluated at the first linearization point [41]. Hence, FEJ would be employed

to avoid inconsistency in EKF estimation. However, the inconsistency effect is a significant issue in filters where the landmarks will be estimated persistently (like in many visual SLAM systems [42]), or the pose states will be kept for a short period (like in MSCKF/SWF cases [43], [44]). For the IEKF in this paper, the current state will only exist in the update stage and will be marginalized instantly after the current IEKF iteration ends. Therefore, FEJ is only meaningful in the current iterations, so inconsistency is not a big issue in our LIO system. The effect of FEJ is tested in our first experiment and will be turned off by default in the next experiments.

V. Experiments

A. Synthetic Data

First, we use the synthetic data to test the point cloud alignment accuracy and convergence speed. The point cloud used in this section is selected from the EPFL statue repository, which contains real-world point clouds reconstructed from photos[1]. Next, we will generate a random pose affected by Gaussian noise (with rotation noise of $\sigma_r = 5$ deg and translation noise of $\sigma_t = 0.1$m) and use this pose to transform the target cloud. Then we test the point cloud alignment performance using the IEKF observation module. The IEKF will start from an identical pose and iteratively converge to the correct pose. Since the ground-truth pose is known, we can record the pose error in each iteration using the equation here:

$$\mathbf{r} = \mathrm{Log}(\mathbf{R}_g^T \mathbf{R}_k) + (\mathbf{t}_g - \mathbf{t}_k), \tag{30}$$

where $\mathbf{R}_g, \mathbf{t}_g$ are ground-truth pose and $\mathbf{R}_k, \mathbf{t}_k$ are the estimated pose in the k-th iteration. The norm of pose error can be used as an indicator of the convergence speed and is shown in Fig. 2. The point cloud model includes "aquaris", "monkeys," "kneeling lady," and "archer." We can see that the IEKF with AA achieves a faster convergence speed in these experiments. The results are similar to that mentioned in pure point cloud registration papers like [25], [29], except that the state variable is a high-dimensional error state rather than a 6 DoF pose.

B. Cost Decrease in Each Iteration

Second, we investigate the lidar point cloud residual distribution during the raw IEKF and AA-IEKF iterations in LIO. However, unlike pure point cloud registration research, the LIO incrementally constructs the point cloud map starting from roughly correct initial guesses. The constructed local map will change with the pose estimation and will also be used as the target point cloud in future estimates. Therefore, when comparing the results with or without AA, the filters are always set to the same starting point. In each align step, we test the AA-IEKF update, roll back to the initial guess, and then update the raw IEKF observation model. We

[1]https://lgg.epfl.ch/statues_dataset.php

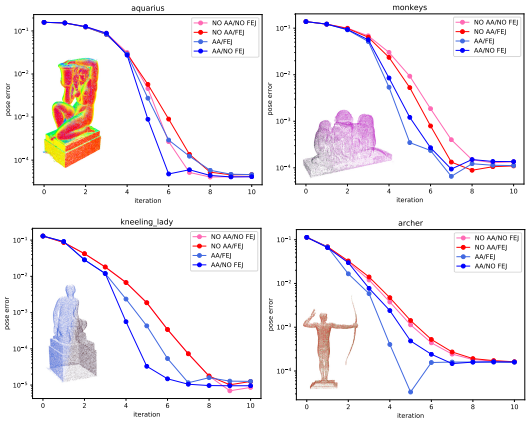

Fig. 2. The ground-truth pose residual decrease with relation to the number of iterations in EPFL dataset.

record the median of the lidar point cloud residuals in each IEKF iteration and normalize them by dividing the initial residual. Hence, both the AA-IEKF and raw IEKF will always start from the same point cloud with the initial residual equal to one. We set the maximum iterations to 10 in the experiments.

The datasets are selected from AVIA [3], NCLT [45], ULHK [46], UTBM robocar dataset [47], and LIO-SAM [48], containing hand-held solid-state lidar and spinning lidar point clouds from self-driving vehicles.

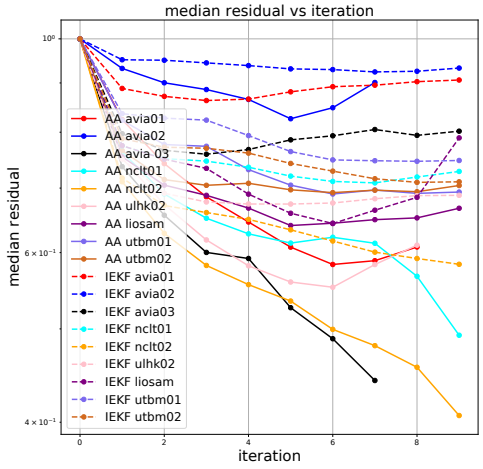

Fig. 3. The median point cloud residual distribution with relation to the number of iteration. Solid lines: AA residuals; Dashed lines: IEKF residuals.

Fig. 3 shows the median residual of each iteration in the sequences mentioned above. Furthermore, Fig. 4 shows the squared error distribution from iterations 0 to 4 of sequence NCLT02. Both results show that AA-IEKF typically achieves faster convergence speed and a lower final cost. The algorithm usually converges in three or four iterations. But the error may still decrease after nine iterations in some sequences. Note that the residual

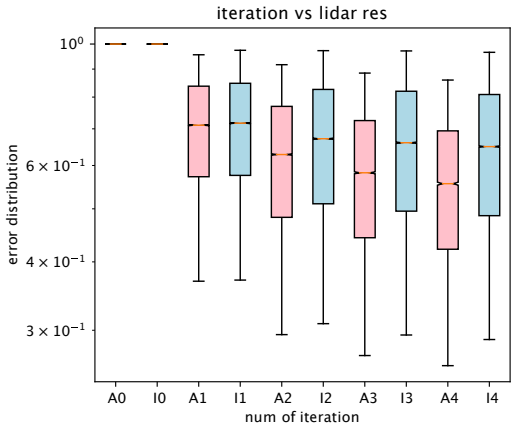

Fig. 4. The squared error distribution in each iteration of NCLT02 sequence. The An denotes the n-th iteration of AA-IEKF and In for IEKF.

may also grow due to the outliers, and the no-converging sequences may have larger residuals in later iterations.

C. Average Number of Iterations in LIO

In the second experiment, we test the average iterations in AA-IEKF and raw IEKF. Using the convergence condition described in Algorithm 1, we compare the number of iterations in each sequence in the dataset. We record the average iterations from the starting point to the optimal solution. The maximum number of iterations is also set to 10 in this experiment (see Fig. 5). It can be seen that Anderson acceleration can effectively reduce the number of iterations in optimization. For solid-state datasets, the AA-IEKF almost requires two iterations to converge, and for spinning lidar datasets, the number of iterations is slightly larger than that.

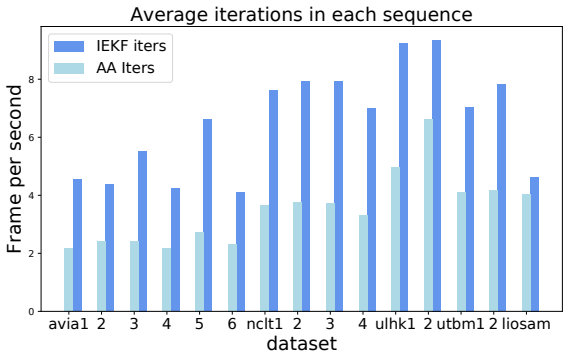

Fig. 5. Average iterations in each sequence.

D. Accuracy Evaluation

This section evaluates the LIO's accuracy in terms of the absolute pose error (APE) and relative pose error (RPE) in 100m with several open-source state-of-the-art LIO algorithms, including FastLIO2 [3], [4], LIO-SAM [48], and LiLi-OM [49]. We adopt the LIO framework from our previous work [5] by replacing the IEKF iterations with AA-IEKF. In the accuracy and time usage evaluation, we set the maximum iterations in IEKF to 10 while keeping other parameters unchanged. The experiments are conducted on a desktop with an AMD R7-5800X CPU and 64 GB memory. The dataset is selected from NCLT [45], UTBM [47], ULHK [46], and LIO-SAM [48]. The loop closing function is turned off to evaluate the odometry performance fairly.

E. Time Usage Evaluation

Finally, we evaluate the time cost in AA-IEKF. As shown before, AA will reduce the average iterations in IEKF, but AA itself will also bring extra computation costs to obtain an AA step. We record the processing time of each scan from sequences in different datasets. The results are displayed in Table II. Note that LIO-SAM and LiLi-OM are keyframe-based LIO systems that employ distributed odometry and optimization nodes. In that case, we will compute the processing time by adding the odometry and optimization time together. For IEKF and AA-IEKF, we set the maximum iterations to ten to make the algorithm fully converge, but the maximum iterations in FastLIO2 are still set to 4 as default[2].

The results show that when the maximum iterations are enlarged, IEKF is generally slower than FastLIO2, even with the acceleration from the better nearest neighbor structure in [5]. But AA-IEKF generally outperforms the other approaches since the average iterations in AA-IEKF are smaller. The LIO-SAM campus sequences are exceptional, where IEKF and AA-IEKF perform almost the same because the algorithm converges quickly (2-3 iterations) in these sequences. In summary, the AA-IEKF can run at more than 150 Hz on a laptop-level CPU platform (like i7-10750H).

VI. Conclusion

In this paper, we present an Anderson-accelerate approach for a general on-manifold error state Kalman filter and apply it in an LIO system. The Anderson acceleration is performed in the tangent space of the starting point in IEKF, preventing additional transforms from the manifold to its Lie algebra. The AA-IEKF can effectively reduce the number of necessary iterations and obtain a low final cost in the point cloud alignment while keeping the time usage in a reasonable range. We think our approach can also be adopted in other IEKF-based systems and can be helpful for future studies.

Acknowledgments

This work in this paper is supported by Beijing Municipal Science and Technology Program (Z211100004221006) and Beijing Nova Program

[2]Note that FastLIO2 has tricks in implementation. It will not update the point cloud nearest neighbor unless the current pose estimate converges. So the nearest neighbor search will usually be computed twice in four iterations, which makes it a bit unfair compared with other algorithms.

TABLE I

Accuracy Evaluation in RPE (% drift per 100 meters) and APE (m)

Map ID	AA-IEKF (%/m)	IEKF (%/m)	FastLio2 (%/m)	LIO-SAM (%/m)	LiLi-OM (%/m)
nclt 1	0.524/1.89	0.491/1.93	0.448/1.75	-	-
nclt 2	0.386/0.95	0.370/0.94	0.347/0.91	0.430/1.11	-
nclt 3	0.392/1.65	0.394/1.94	0.346/2.12	0.366/5.83	-
nclt 4	0.481/1.31	0.513/1.32	0.408/0.82	-	-
utbm 1	0.665/13.7	0.666/14.5	0.714/12.7	-	2.261/63.1
utbm 2	0.828/14.7	0.830/15.1	0.836/13.3	-	1.199/82.0
utbm 3	0.443/14.1	0.444/14.8	0.481/14.6	-	1.800/102.3
utbm 4	1.523/7.59	1.524/7.77	1.797/7.22	-	1.319/48.0
utbm 5	0.273/14.4	0.273/15.2	0.295/12.9	-	1.995/76.2
utbm 6	1.176/13.7	1.181/14.2	1.184/13.3	-	6.718/77.9
ulhk 1	2.086/0.65	2.186/1.24	1.477/1.20	1.363/1.48	1.877/9.98
ulhk 2	1.700/0.65	1.701/1.13	1.624/1.10	1.369/1.81	1.414/10.3
liosam campus small	1.014/0.81	1.106/1.77	0.751/0.83	0.654/0.47	-

[1] We do not adjust the parameter settings for LIO-SAM and LiLi-OM, which may cause the older algorithms to not run perfectly in newer datasets.

[2] "-" means the algorithm failed in this sequence due to large drift or lack of necessary input data.

TABLE II

Processing Time of Each Scan

Map ID	AA-IEKF (ms)	IEKF (ms)	FastLio2 (ms)	LIO-SAM (ms)	LiLi-OM (ms)
nclt 1	6.26	9.67	6.59	70.77	25.36
nclt 2	4.75	7.07	7.90	42.19	30.65
nclt 3	6.27	9.56	6.29	61.62	16.38
utbm 1	6.12	7.94	9.48	-	35.25
utbm 2	6.48	8.06	8.60	-	33.43
utbm 3	6.49	8.12	8.58	-	33.45
ulhk 1	5.13	9.61	6.90	26.37	25.9
liosam campus small	3.13	3.10	7.99	48.47	-
liosam campus large	3.51	3.45	6.51	50.88	-

[1] The maximum number of iterations in AA-IEKF and IEKF is set to 10, while in FastLIO2 it is set to 4 as default.

(Z201100006820047) from Beijing Municipal Science and Technology Commission.

Appendix

In the appendix section, we give the exact forms of the dynamics and the Jacobian matrices mentioned in the previous sections.

A. Nominal State and Error State Dynamics

The nominal state is propagated by the gyroscope and accelerator inputs: $\mathbf{u}(k) = [\boldsymbol{\omega}, \mathbf{a}]^T(k)$, which is affected by the IMU noise: $\mathbf{w}(k) = [\mathbf{n}_g, \mathbf{n}_a, \mathbf{n}_{bg}, \mathbf{n}_{ba}]^T$.

The prediction equations of the nominal state are:

$$\begin{aligned}\mathbf{x}_{\text{pred}}(k+1) &= \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \\ &= \begin{bmatrix} \mathbf{p}(k) + \mathbf{v}(k)\Delta t \\ \mathbf{R}(k)\text{Exp}((\boldsymbol{\omega} - \mathbf{b}_g)(k)\Delta t) \\ \mathbf{v}(k) + (\mathbf{R}(k)(\mathbf{a} - \mathbf{b}_a(k)) + \mathbf{g}(k))\Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \tag{31}$$

The error state dynamics are same with [13]:

$$\begin{aligned}&\delta\mathbf{x}_{\text{pred}}(k+1) = \mathbf{f}'(\delta\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)) = \\ &\begin{bmatrix} \delta\mathbf{p} + \delta\mathbf{v}\Delta t \\ \text{Exp}(\ (\boldsymbol{\omega} \quad \mathbf{b}_g)\Delta t)\delta\boldsymbol{\theta} \quad \mathbf{J}_r((\boldsymbol{\omega} \quad \mathbf{b}_g)\Delta t)\delta\mathbf{b}_g\Delta t + \mathbf{n}_g\Delta t \\ \delta\mathbf{v} + (\ \mathbf{R}(\mathbf{a} \quad \mathbf{b}_a)^\wedge\delta\boldsymbol{\theta} \quad \mathbf{R}\delta\mathbf{b}_a \quad \mathbf{g}^\wedge\mathbf{B}(\mathbf{g})\delta\mathbf{g})\Delta t + \mathbf{n}_a\Delta t \\ \delta\mathbf{b}_g + \mathbf{n}_{bg}\Delta t \\ \delta\mathbf{b}_a + \mathbf{n}_{ba}\Delta t \\ \delta\mathbf{g} \end{bmatrix}, \end{aligned} \tag{32}$$

where $\mathbf{J}_r(\mathbf{x})$ is the right Jacobian matrix in SO(3):

$$\begin{aligned}\mathbf{J}_r(\mathbf{x}) = \frac{\sin(||\mathbf{x}||)}{||\mathbf{x}||}\mathbf{I} + (1 - \frac{\sin(||\mathbf{x}||)}{||\mathbf{x}||})\frac{\mathbf{x}\mathbf{x}^T}{||\mathbf{x}||^2} \\ + \frac{1 - \cos(||\mathbf{x}||)}{||\mathbf{x}||}\frac{\mathbf{x}^\wedge}{||\mathbf{x}||}. \end{aligned} \tag{33}$$

B. Jacobian Matrix of the Motion Equation

The Jacobian matrix of the motion equation is just the linearized form of (32):

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & 0 & \mathbf{I}\Delta t & 0 & 0 & 0 \\ 0 & \mathbf{F}_{\text{RR}} & 0 & \mathbf{F}_{\text{Rb}_g} & 0 & 0 \\ 0 & \mathbf{F}_{\text{vR}} & \mathbf{I} & 0 & -\mathbf{R}\Delta t & \mathbf{F}_{\text{vg}} \\ 0 & 0 & 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{I} \end{bmatrix}, \tag{34}$$

where

$$\mathbf{F}_{\mathrm{RR}} = \mathrm{Exp}(-(\boldsymbol{\omega} - \mathbf{b}_g)\Delta t), \tag{35}$$

$$\mathbf{F}_{\mathrm{Rb}_g} = -\mathbf{J}_r((\boldsymbol{\omega} - \mathbf{b}_g)\Delta t)\Delta t, \tag{36}$$

$$\mathbf{F}_{\mathrm{vR}} = -\mathbf{R}(a - \mathbf{b}_a)^{\wedge}\Delta t, \tag{37}$$

$$\mathbf{F}_{\mathrm{vg}} = -(\mathbf{g}^{\wedge})\mathbf{B}(\mathbf{g})\Delta t. \tag{38}$$

The Jacobian matrix of S(2) is:

$$\mathbf{B}\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}\right) = \frac{1}{l}\begin{bmatrix} -b & -c \\ l - bb/(l+a) & -bc/(l+a) \\ -bc/(l+a) & l - cc/(l+a) \end{bmatrix}, \tag{39}$$

where $l = \sqrt{a^2 + b^2 + c^2}$.

The $\mathbf{M}(\mathbf{x}, \delta\mathbf{x})$ in S(2) is given as:

$$\mathbf{M}(\mathbf{x}, \delta\mathbf{x}) = -\mathrm{Exp}(\mathbf{B}(\mathbf{x})\delta\mathbf{x})\mathbf{x}^{\wedge}\mathbf{J}_r(\mathbf{B}(\mathbf{x})\delta\mathbf{x})\mathbf{B}(\mathbf{x}). \tag{40}$$

C. Jacobian Matrix of the Observation Equation

For a lidar point $\mathbf{p}_i$ and its nearest neighbor, we denote the corresponding plane $\mathbf{s}_i$ by a norm vector $\mathbf{n}_i$ and a center $\mathbf{q}_i$. Then point-to-plane distance d_i can be calculated by the residual function:

$$d_i = h(\mathbf{x}) = \mathbf{n}_i^T(\mathbf{R}\mathbf{p}_i + \mathbf{p} - \mathbf{q}_i). \tag{41}$$

The partial derivatives of the states are:

$$\mathbf{H}(\delta\boldsymbol{\theta}) = \frac{\partial h}{\partial \mathbf{R}}\frac{\partial \mathbf{R}}{\partial \delta\boldsymbol{\theta}} = -\mathbf{n}_i^T\mathbf{R}\mathbf{p}_i^{\wedge}, \tag{42}$$

$$\mathbf{H}(\delta\mathbf{p}) = \frac{\partial h}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \delta\mathbf{p}} = \mathbf{n}_i^T. \tag{43}$$

References

[1] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," IEEE Transactions on robotics and automation, vol. 17, no. 3, pp. 229–241, 2001.

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 6, pp. 1052–1067, 2007.

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

[4] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," arXiv preprint arXiv:2107.06829, 2021.

[5] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels," IEEE Robotics and Automation Letters, pp. 1–1, 2022.

[6] S. Thrun, "Probabilistic robotics," Communications of the ACM, vol. 45, no. 3, pp. 52–57, 2002.

[7] T. D. Barfoot, State estimation for robotics. Cambridge University Press, 2017.

[8] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," IEEE Transactions on robotics, vol. 23, no. 5, pp. 1036–1049, 2007.

[9] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3562–3568, IEEE, 2006.

[10] M. Deans and M. Hebert, "Experimental comparison of techniques for localization and mapping using a bearing-only sensor," in Experimental Robotics VII, pp. 395–404, Springer, 2001.

[11] W. Wen, T. Pfeifer, X. Bai, and L.-T. Hsu, "Factor graph optimization for gnss/ins integration: A comparison with the extended kalman filter," NAVIGATION, Journal of the Institute of Navigation, vol. 68, no. 2, pp. 315–331, 2021.

[12] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in Robotics: Science and Systems, vol. 2, pp. 1–9, Berkeley, CA, 2014.

[13] J. Sola, "Quaternion kinematics for the error-state kalman filter," arXiv preprint arXiv:1711.02508, 2017.

[14] V. Madyastha, V. Ravindra, S. Mallikarjunan, and A. Goyal, "Extended kalman filter vs. error state kalman filter for aircraft attitude estimation," in AIAA Guidance, Navigation, and Control Conference, p. 6615, 2011.

[15] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," The International Journal of Robotics Research, vol. 36, no. 10, pp. 1053–1072, 2017.

[16] A. I. Mourikis, S. I. Roumeliotis, et al., "A multi-state constraint kalman filter for vision-aided inertial navigation.," in ICRA, vol. 2, p. 6, 2007.

[17] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," IET Control Theory & Applications, vol. 4, no. 8, pp. 1303–1318, 2010.

[18] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual slam: why filter?," Image and Vision Computing, vol. 30, no. 2, pp. 65–77, 2012.

[19] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," arXiv preprint arXiv:2102.03804, 2021.

[20] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," IEEE Transactions on Robotics, vol. 33, no. 1, pp. 1–21, 2016.

[21] S.-F. Ch'ng, A. Khosravian, A.-D. Doan, and T.-J. Chin, "Outlier-robust manifold pre-integration for ins/gps fusion," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7489–7496, IEEE, 2019.

[22] S. Bouaziz, A. Tagliasacchi, H. Li, and M. Pauly, "Modern techniques and applications for real-time non-rigid registration," in SIGGRAPH ASIA 2016 Courses, pp. 1–25, 2016.

[23] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in Sensor fusion IV: control paradigms and data structures, vol. 1611, pp. 586–606, Spie, 1992.

[24] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," arXiv preprint arXiv:2102.03804, 2021.

[25] J. Zhang, Y. Yao, and B. Deng, "Fast and robust iterative closest point," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[26] D. G. Anderson, "Iterative procedures for nonlinear integral equations," Journal of the ACM (JACM), vol. 12, no. 4, pp. 547–560, 1965.

[27] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu, "Geometry and convergence analysis of algorithms for registration of 3d shapes," International Journal of Computer Vision, vol. 67, no. 3, pp. 277–296, 2006.

[28] S. Rusinkiewicz, "A symmetric objective function for icp," ACM Transactions on Graphics (TOG), vol. 38, no. 4, pp. 1–7, 2019.

[29] A. L. Pavlov, G. W. Ovchinnikov, D. Y. Derbyshev, D. Tsetserukou, and I. V. Oseledets, "Aa-icp: Iterative closest point with anderson acceleration," in 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 3407–3412, IEEE, 2018.

[30] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," IEEE transactions on Robotics, vol. 23, no. 1, pp. 34–46, 2007.

[31] L. M. Paz, J. D. Tardós, and J. Neira, "Divide and conquer: Ekf slam in $o(n)$," IEEE Transactions on Robotics, vol. 24, no. 5, pp. 1107–1120, 2008.

[32] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A first-estimates jacobian ekf for improving slam consistency," in Experimental Robotics, pp. 373–382, Springer, 2009.

[33] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremen-

tal smoothing and mapping," IEEE Transactions on Robotics, vol. 24, no. 6, pp. 1365–1378, 2008.
[34] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3184–3191, 2021.
[35] L. Zhou, D. Koppel, and M. Kaess, "Lidar slam with plane adjustment for indoor environment," IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 7073–7080, 2021.
[36] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4758–4765, IEEE, 2018.
[37] H. Pottmann, S. Leopoldseder, and M. Hofer, "Registration without icp," Computer Vision and Image Understanding, vol. 95, no. 1, pp. 54–71, 2004.
[38] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," in Computer graphics forum, vol. 32, pp. 113–123, Wiley Online Library, 2013.
[39] J. Zhang, B. O'Donoghue, and S. Boyd, "Globally convergent type-i anderson acceleration for nonsmooth fixed-point iterations," SIAM Journal on Optimization, vol. 30, no. 4, pp. 3170–3197, 2020.
[40] C. Hertzberg, "A framework for sparse, non-linear least squares problems on manifolds," in Universität Bremen, Citeseer, 2008.
[41] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A first-estimates jacobian ekf for improving slam consistency," in Experimental Robotics, pp. 373–382, Springer, 2009.
[42] S. Jia, Y. Jiao, Z. Zhang, R. Xiong, and Y. Wang, "Fej-viro: A consistent first-estimate jacobian visual-inertial-ranging odometry," arXiv preprint arXiv:2207.08214, 2022.
[43] A. I. Mourikis, S. I. Roumeliotis, et al., "A multi-state constraint kalman filter for vision-aided inertial navigation.," in ICRA, vol. 2, p. 6, 2007.
[44] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 3, pp. 611–625, 2017.
[45] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan North Campus long-term vision and lidar dataset," International Journal of Robotics Research, vol. 35, no. 9, pp. 1023–1035, 2015.
[46] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu, "Urbanloco: a full sensor suite dataset for mapping and localization in urban scenes," in 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2310–2316, IEEE, 2020.
[47] Z. Yan, L. Sun, T. Krajnik, and Y. Ruichek, "EU long-term dataset with multiple sensors for autonomous driving," in Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
[48] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5135–5142, IEEE, 2020.
[49] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping," IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 5167–5174, 2021.