Generalized n-Dimensional Rigid Registration: Theory and Applications

Jin Wu, *Member, IEEE*, Miaomiao Wang, *Member, IEEE*, Hassen Fourati, Hui Li, *Member, IEEE*, Yilong Zhu, Chengxi Zhang, *Member, IEEE*, Yi Jiang, *Member, IEEE*, Xiangcheng Hu, and Ming Liu, *Senior Member, IEEE*

***Abstract*—The generalized rigid registration problem in high-dimensional Euclidean spaces is studied. The loss function is minimized with an equivalent error formulation by the Cayley formula. The closed-form linear least-square solution to such a problem is derived which generates the registration covariances, i.e., uncertainty information of rotation and translation, providing quite accurate probabilistic descriptions. Simulation results indicate the correctness of the proposed method and also present its efficiency on computation-time consumption, compared with previous algorithms using singular value decomposition (SVD) and linear matrix inequality (LMI). The proposed scheme is then applied to an interpolation problem on the special Euclidean group $SE(n)$ with covariance-preserving functionality. Finally, experiments on covariance-aided Lidar mapping show practical superiority in robotic navigation.**

***Index Terms*—Covariance analysis, navigation, point-cloud registration, rigid transformation, robotic perception.**

I. Introduction

A. Background and Related Works

POINT-CLOUD registration has been extensively developed in the past few decades and widely employed in various fields, including robotic perception, automated reconstruction, computer-aided design (CAD), etc. [1]–[3]. The points can either be measured by a 2-D/3-D laser scanner, or by a time-of-flight (ToF) sensor, or even by scene reconstruction from monocular/binocular cameras [4], [5]. The basic purpose of the point-cloud registration is to find out the rigid/affine/nonrigid transformations between two measured point sets, while most registration problems can be considered as locally rigid ones as much as possible [6], [7]. By virtue of this aim, many engineering processes require the point-cloud registration, including spacecraft attitude determination, autonomous navigation, simultaneous localization and mapping (SLAM), etc. [8]–[12]. Thus, the principle of point-cloud registration is to seek the most appropriate correspondences along with an optimal transformation to match the two-point sets. Note that the point numbers of the two sets do not have to be consistent, so such a problem is usually nonconvex during matching. For rigid point-cloud registration, the best correspondences and the optimal rigid transformation are usually unified with the iterative closest point (ICP, [13]). In ICP, the correspondences can be figured out iteratively by means of brute-force searching or aided by an kD tree, with a given transformation guess. The rigid transformation is often obtained via closed-form solutions, e.g., singular value decomposition (SVD, [14], [15]), eigen-decomposition (EIG, [16]), dual quaternion method [17], etc. [18]. ICP is practical and efficient but suffers from local minima during iterations. Therefore, some efforts have been paid to find the global optimum of such a problem, e.g., the Go-ICP [19]. Mainstream ICP variants mainly deal with the 3-D registration case while the high-dimensional one is actually needed for some other cases, such as the localization of sensor networks [20], [21].

The main work presented in this article is to show the closed-form solution and its covariance analysis of such a high-dimensional rigid registration problem. To describe the shape of rigid transformations, special orthogonal groups, and special Euclidean groups are usually invoked [22], [23]. Studying the control problems based on such manifolds has become popular in recent years, involving some results on feedback control laws, optimization hull, motion planning, etc. [24]–[26]. The specific problem of n-dimensional rigid registration can be solved via the SVD or linear matrix inequality (LMI) [27], [28]. However, as both these methods suffer from the existence of high nonlinearity, a covariance analysis guaranteeing reliable quality control may be not feasible at the current stage.

This work was supported in part by the Shenzhen Science, Technology and Innovation Commission under Grant JCYJ20160401100022706, and in part by the National Natural Science Foundation of China under Grant U1836218.

Jin Wu, Yilong Zhu, Xiangcheng Hu, and Ming Liu are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: eelium@ust.hk).

Miaomiao Wang is with the Department of Electronic and Computer Engineering, University of Western Ontario, London, ON N6A 3K7, Canada (e-mail: mwang448@uwo.ca).

Hassen Fourati is with the University Grenoble Alpes, CNRS, GIPSA-Lab, 38400 Grenoble, France, and also with Inria, Grenoble, France (e-mail: hassen.fourati@gipsa-lab.grenoble-inp.fr).

Hui Li is with the School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214126, China (e-mail: lihui.cv@jiangnan.edu.cn).

Chengxi Zhang is with the School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: dongfangxy@163.com).

Yi Jiang is with the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong (e-mail: yjian22@cityu.edu.hk).

The engineering background for the presented study is that, apart from those nonlinear observers on the 3-D special orthogonal groups [29], [30], higher dimensional registration techniques have been ultilized by the authors in [31] to solve the robotic hand-eye calibration between the robotic gripper and attached camera, of the type $\boldsymbol{AX} = \boldsymbol{XB}$ with $\boldsymbol{A}, \boldsymbol{B}$ being known and $\boldsymbol{X}$ being unknown. It is revealed that a mapping from 3-D special Euclidean group to the 4-D special orthogonal group will be of convenience in solving such problem. In the 4-D case, the authors propose the unit octonion method for point-cloud registration but this is not extendable regarding registration with arbitrary dimensions. As hand-eye calibration has received extensive research during the past several decades [32]–[36], the proposed n-dimensional case may benefit future related works. It is also noticed that, for the 3-D case, Barczyk *et al.* [37] have derived the closed form of the ICP matching covariance and the results have been later fused with the inertial measurement using an invariant Kalman filter. As mentioned above, the 3-D and 4-D cases are all specific ones that can not give mandatory information for the extension to n-dimensional ones. Therefore, the main challenge confronted is to find an efficient universal parameterization approach for the registration on the n-dimensional Euclidean space.

B. Contributions

Following above problems in n-dimensional registration, this article proposes a new formulation as a linear solution. Major contributions are listed as follows.

1) Based on the Caylay transformation, linear results regarding the registration problem have been derived. The developed method provides a new perspective other than existing ones like SVD.
2) Computational burden has been significantly decreased by simplifying related computation steps via specific matrix manipulations. With this technique, the online efficiency of the algorithm has been improved.
3) The uncertainty descriptions of the derived solution are also derived which gives the quantization of the quality of registration with given noisy point clouds.

Following these contributions, through simulation and experimental results, we also show the advantages of the proposed method in algorithmic implementation and efficiency, compared with representatives, such as LMI, SVD, and other applications.

C. Outline

This article is structured as follows: Section II presents the problem formulation and introduces our proposed linear solution and covariance analysis. Section III consists of experiments, results, and comparisons while concluding remarks are drawn in Section IV.

D. Notations

The n-dimensional Euclidean vector space is described with $\mathbb{R}^n$. We use $\mathbb{R}^{n\times m}$ to denote the real space containing all matrices with row dimension of n and column dimension of m. The identity matrix has the notation of $\boldsymbol{I}$ and owns a certain size according to the context. $\boldsymbol{X}^\top$ and $\boldsymbol{X}^{-1}$ mean the transpose and inverse of a given matrix $\boldsymbol{X}$, respectively, in which the inverse exists when $\boldsymbol{X}$ is square and nonsingular. We use tr to represent the trace of a square matrix. The adj denotes the adjoint matrix. $\|\cdot\|$ stands for the l_2 norm in the Euclidean space such that $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^\top \boldsymbol{x}}$ for any given column vector $\boldsymbol{x}$. rank($\boldsymbol{X}$) depicts the row rank information of $\boldsymbol{X}$. For the 3-D Lie algebra, the special orthogonal group $SO(3)$ contains all the orthonormal rotation matrices in $\mathbb{R}^{3\times 3}$. It is extended to n-dimensional Euclidean space with the $SO(n)$ whose identity is expressed with $\boldsymbol{X} \in SO(n) \Rightarrow \boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{I}$, $\det(\boldsymbol{X}) = 1$. The $\mathfrak{so}(3)$ contains all skew-symmetric matrices $\boldsymbol{x}_\times$ from any 3-D vector $\boldsymbol{x} = (x_1, x_2, x_3)^\top$ such that the cross product between any two 3-Dl vectors $\boldsymbol{x}, \boldsymbol{y}$ is equivalent to $\boldsymbol{x}\times\boldsymbol{y} = \boldsymbol{x}_\times\boldsymbol{y} = -\boldsymbol{y}_\times\boldsymbol{x}$. The generalization of $\mathfrak{so}(3)$ from 3-D space to the n-D space is $\mathfrak{so}(n)$. Note that any element on the group $\mathfrak{so}(n)$ is a skew-symmetric matrix and can be exponentially mapped to a unique rotation matrix on $SO(n)$. With a $([n(n-1)]/2)$-D vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_{([n(n-1)]/2)}]^\top$, the associated skew-symmetric matrix is defined in (1) so that $\boldsymbol{x}_\times \in \mathfrak{so}(n)$

$$\boldsymbol{x}_\times = \begin{pmatrix} 0 & -x_{\frac{n(n-1)}{2}} & x_{\frac{n(n-1)}{2}-1} & \cdots & (-1)^{n-2}x_{2n-3} & (-1)^{n-1}x_{n-1} \\ * & 0 & -x_{\frac{n(n-1)}{2}-2} & \cdots & (-1)^{n-3}x_{2n-4} & (-1)^{n-2}x_{n-2} \\ * & * & \ddots & \cdots & \vdots & \vdots \\ * & * & * & \ddots & -x_n & x_2 \\ * & * & * & * & 0 & -x_1 \\ * & * & * & * & * & 0 \end{pmatrix}. \tag{1}$$

The inverse map, i.e., the wedge operation $\wedge$ from the $n \times n$ skew-symmetric matrix to the $([n(n-1)]/2)$-D vector is denoted as $\boldsymbol{x}_\times^\wedge = \boldsymbol{x}$

II. High-Dimensional Registration: Solution and Covariances

A. Problem Formulation

The generalized rigid registration problem can be characterized with the following optimization [13]:

$$\underset{\boldsymbol{R}\in SO(n), \boldsymbol{t}\in\mathbb{R}^3}{\arg\min} \quad \mathcal{L} = \sum_{i=1}^{N} w_i \left\| \boldsymbol{b}_i - \boldsymbol{R}\boldsymbol{r}_i - \boldsymbol{t} \right\|^2 \tag{3}$$

in which $\boldsymbol{R}$ is the n-dimensional rotation matrix; $\boldsymbol{t}$ represents the Euclidean translation vector in the $\mathbb{R}^n$. The rotation and translation, together, namely, the homogeneous transformation $\begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \in SE(n)$ in the special Euclidean group $SE(n)$, relates N vector pairs $\{\boldsymbol{b}_i, \boldsymbol{r}_i | i = 1, 2, \ldots, N\}$ in the body frame b and reference frame r together. Here, the relationship between vector pairs is expressed with the normalized weight w_i, such that $\sum_{i=1}^{N} w_i = 1$. The weights actually denote the uncertainty characteristics of the data to be aligned. However, in practice, when there are large amount of points, e.g., for the Lidar mapping, there is no criteria to determine the weights. In that case, the weights are computed

equally as $w_i = 1/N$. The problem (3) will always be convex by introducing the unit quaternion for representation of $\boldsymbol{R}$ in $SO(3)$ [38]. The uniqueness of the problem can be found in [15]. The problem is a least-square one and can be solved with many techniques [39]. When the dimension increases, the quaternion will be no longer feasible to give adequate description of rotations. Let us think about the general rotation representation and factorization that is independent of the dimension n. For any orthonormal rotation matrix with dimension of three, one has $\boldsymbol{R}\boldsymbol{\xi} = \boldsymbol{\xi}$, where $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)^\top$ is called the eigenaxis of $\boldsymbol{R}$. While for the rotation in $SO(3)$, the Rodrigues formula can be accordingly derived as $\boldsymbol{R} = \cos\theta\boldsymbol{I} + (1-\cos\theta)\boldsymbol{\xi}\boldsymbol{\xi}^\top + \sin\theta\boldsymbol{\xi}_\times$, where θ is the intermediate rotation angle about the eigenaxis $\boldsymbol{\xi}$.

The high-dimensional extension of Rodrigues formula is in a sophisticated form since the cross-product of vectors are only proven to exist in three and seven dimensional Euclidean spaces [40]. Moreover, the Rodrigues formula is nonlinear and can hardly offer convenience for rotation computation. However, the Cayley transformation, i.e.,

$$\boldsymbol{R} = (\boldsymbol{I}+\boldsymbol{G})^{-1}(\boldsymbol{I}-\boldsymbol{G}) \tag{4}$$

with $\boldsymbol{G}$ denoting an n-dimensional skew-symmetric matrix, always holds for the rotation factorization. Such technique has been invoked for solving Wahba's problem that is a special case of (3) with $n = 3$, $\boldsymbol{t} = 0$, and $\|\boldsymbol{b}_i\| = \|\boldsymbol{r}_i\| = 1$ forming a spacecraft attitude estimator called OLAE [41]. With different data dimensions $n > 3$, the problem significantly varies. The matrices $\boldsymbol{I}+\boldsymbol{G}$ and $\boldsymbol{I}-\boldsymbol{G}$ are often invertible, but will suffer from singularities when all Euler angles approach $\pm\pi$, which is a special case. However, in engineering, there is almost no such a coincident case for n-dimensional registration, because of the noise of the input data. Therefore, we assume here that $\boldsymbol{I}+\boldsymbol{G}$ and $\boldsymbol{I}-\boldsymbol{G}$ are strictly invertible. In the following parts, we are going to present the proposed linear solution and associated covariance analysis.

B. Proposed Linear Solution

Let us define the centers of the mass of the point sets as

$$\bar{\boldsymbol{b}} = \sum_{i=1}^{N} w_i \boldsymbol{b}_i, \quad \bar{\boldsymbol{r}} = \sum_{i=1}^{N} w_i \boldsymbol{r}_i \tag{5}$$

with $\boldsymbol{b}_i, \boldsymbol{r}_i \in \mathbb{R}^n$. For large numbers of points, without loss of generality, the covariance of each point can be unified with $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\boldsymbol{b}_i} = \boldsymbol{\Sigma}_{\boldsymbol{r}_i}$. Then the rotation-only problem is converted to the following optimization:

$$\arg\min_{\boldsymbol{R}\in SO(n)} \mathcal{L} = \sum_{i=1}^{N} w_i \left\| \boldsymbol{b}_i - \bar{\boldsymbol{b}} - \boldsymbol{R}(\boldsymbol{r}_i - \bar{\boldsymbol{r}}) \right\|^2. \tag{6}$$

With the virtue of (4) and defining $\tilde{\boldsymbol{b}}_i = \boldsymbol{b}_i - \bar{\boldsymbol{b}}, \tilde{\boldsymbol{r}}_i = \boldsymbol{r}_i - \bar{\boldsymbol{r}}$, and the error vector $\boldsymbol{e}_i = \tilde{\boldsymbol{b}}_i - \boldsymbol{R}\tilde{\boldsymbol{r}}_i$, one always expect the error to be zero, such that

$$\boldsymbol{e}_i = \tilde{\boldsymbol{b}}_i - (\boldsymbol{I}+\boldsymbol{G})^{-1}(\boldsymbol{I}-\boldsymbol{G})\tilde{\boldsymbol{r}}_i = \boldsymbol{0}. \tag{7}$$

The right item can be further expressed as

$$\begin{aligned} \tilde{\boldsymbol{b}}_i &= (\boldsymbol{I}+\boldsymbol{G})^{-1}(\boldsymbol{I}-\boldsymbol{G})\tilde{\boldsymbol{r}}_i \Rightarrow (\boldsymbol{I}+\boldsymbol{G})\tilde{\boldsymbol{b}}_i = (\boldsymbol{I}-\boldsymbol{G})\tilde{\boldsymbol{r}}_i \\ &\Rightarrow \boldsymbol{G}\left(\tilde{\boldsymbol{b}}_i + \tilde{\boldsymbol{r}}_i\right) = \tilde{\boldsymbol{r}}_i - \tilde{\boldsymbol{b}}_i. \end{aligned} \tag{8}$$

Note that in [41], the above equation can be further derived to

$$\left(\tilde{\boldsymbol{b}}_i + \tilde{\boldsymbol{r}}_i\right)_\times \boldsymbol{g} = \tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{r}}_i \tag{9}$$

where $\boldsymbol{G}$ is replaced by $\boldsymbol{G} = \boldsymbol{g}_\times \in \mathfrak{so}(3)$. The equation can be obtained by symbolic manipulation using MATLAB, Maple or Mathematica. That is to say the minimization (6) is transformed into

$$\arg\min_{\boldsymbol{G}^\top = -\boldsymbol{G}} \mathcal{L} = \sum_{i=1}^{N} w_i \|\boldsymbol{G}\boldsymbol{x}_i - \boldsymbol{d}_i\|^2 \tag{10}$$

where

$$\begin{aligned} \boldsymbol{x}_i &= \tilde{\boldsymbol{b}}_i + \tilde{\boldsymbol{r}}_i \\ \boldsymbol{d}_i &= \tilde{\boldsymbol{r}}_i - \tilde{\boldsymbol{b}}_i. \end{aligned} \tag{11}$$

For the n-dimensional $\boldsymbol{G}$, we need to determine the $n(n-1)/2$ items inside $\boldsymbol{G}$ such that

$$\begin{aligned} \boldsymbol{G}(\boldsymbol{g}) &= \boldsymbol{g}_\otimes \\ &= \begin{pmatrix} 0 & g_1 & g_2 & \cdots & g_{n-1} \\ -g_1 & 0 & g_n & \cdots & g_{2n-3} \\ -g_2 & g_n & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & 0 & g_{\frac{n(n-1)}{2}} \\ -g_{n-1} & -g_{2n-3} & \cdots & -g_{\frac{n(n-1)}{2}} & 0 \end{pmatrix} \end{aligned} \tag{12}$$

with $\boldsymbol{g} = [g_1, g_2, \ldots, g_{([n(n-1)]/2)}]^\top$ and $\boldsymbol{G}$ being a linear function of $\boldsymbol{g}$. It can be noticed that although the linear mapping $\boldsymbol{g}_\otimes$

$$\boldsymbol{P}_4(\boldsymbol{x}_i) = \begin{pmatrix} x_{i,2} & x_{i,3} & x_{i,4} & 0 & 0 & 0 \\ -x_{i,1} & 0 & 0 & x_{i,3} & x_{i,4} & 0 \\ 0 & -x_{i,1} & 0 & -x_{i,2} & 0 & x_{i,4} \\ 0 & 0 & -x_{i,1} & 0 & -x_{i,2} & -x_{i,3} \end{pmatrix}$$

$$\boldsymbol{P}_5(\boldsymbol{x}_i) = \begin{pmatrix} x_{i,2} & x_{i,3} & x_{i,4} & x_{i,5} & 0 & 0 & 0 & 0 & 0 & 0 \\ -x_{i,1} & 0 & 0 & 0 & x_{i,3} & x_{i,4} & x_{i,5} & 0 & 0 & 0 \\ 0 & -x_{i,1} & 0 & 0 & -x_{i,2} & 0 & 0 & x_{i,4} & x_{i,5} & 0 \\ 0 & 0 & -x_{i,1} & 0 & 0 & -x_{i,2} & 0 & -x_{i,3} & 0 & x_{i,5} \\ 0 & 0 & 0 & -x_{i,1} & 0 & 0 & -x_{i,2} & 0 & -x_{i,3} & -x_{i,4} \end{pmatrix} \tag{2}$$

is also on $\mathfrak{so}(n)$, it does not require the specific arrangements of elements and signs that are needed inside the mapping from $\boldsymbol{G}$ to $\boldsymbol{g}_\times$ in (1). Therefore, $\boldsymbol{g}_\otimes$ is more flexible than $\boldsymbol{g}_\times$ in representing $\boldsymbol{G}$ (see `skewdec` command in MATLAB software for some details). Then for arbitrary $\boldsymbol{G}\boldsymbol{x}_i$ in (10), we can always find a corresponding matrix $\boldsymbol{P}_n(\boldsymbol{x}_i)$ satisfying $\boldsymbol{G}(\boldsymbol{g})\boldsymbol{x}_i = \boldsymbol{P}_n(\boldsymbol{x}_i)\boldsymbol{g}$. This can be achieved via many concurrent mathematical tools. For instance, in the MATLAB, we can use `equationsToMatrix` function to obtain instant expression of $\boldsymbol{P}_i$. The optimization (10) is then solved by

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{g}} &= \boldsymbol{0}\\ &\Rightarrow \boldsymbol{g}^\top \sum_{i=1}^{N} w_i \boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i) - \sum_{i=1}^{N} w_i \boldsymbol{d}_i^\top \boldsymbol{P}(\boldsymbol{x}_i) = \boldsymbol{0}\\ &\Rightarrow \sum_{i=1}^{N} w_i \boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)\boldsymbol{g} = \sum_{i=1}^{N} w_i \boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{d}_i. \end{aligned} \tag{13}$$

It is obvious that the solution is linear and it follows:

$$\boldsymbol{g} = \boldsymbol{H}^{-1}\boldsymbol{v} \tag{15}$$

with

$$\begin{aligned}\boldsymbol{H} &= \sum_{i=1}^{N} w_i \boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)\\ \boldsymbol{v} &= \sum_{i=1}^{N} w_i \boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{d}_i. \end{aligned} \tag{16}$$

Then, the rotation is reconstructed by

$$\boldsymbol{R} = \left(\boldsymbol{I} + \boldsymbol{g}_\otimes\right)^{-1}\left(\boldsymbol{I} - \boldsymbol{g}_\otimes\right). \tag{17}$$

As a convention, the translation can be determined with $\boldsymbol{t} = \bar{\boldsymbol{b}} - \boldsymbol{R}\bar{\boldsymbol{r}}$ [14].

Now that we have already known $\boldsymbol{P}(\boldsymbol{x}_i) \in \mathbb{R}^{n\times([n(n-1)]/2)}$. The time complexity of the matrix inversion varies from $O(n^{2.373})$ to $O(n^3)$ using the optimized Coppersmith–Winograd method and the Schoolbook multiplication, respectively, [42]. The matrix $\boldsymbol{H} \in \mathbb{R}^{([n(n-1)]/2)\times([n(n-1)]/2)}$ will consume huge load of time to compute its inversion with increasing order $n \to +\infty$. Thus, the emerging task is to figure out a faster matrix inversion of $\boldsymbol{H}$ by dimension reduction through the following properties.

C. Fast Matrix Inversion

The diagonal zero elements of $\boldsymbol{G}$ make $\boldsymbol{P}(\boldsymbol{x}_i)$ a matrix with row-rank of $n-1$, which also leads to

$$\operatorname{rank}\left[\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)\right] = n-1. \tag{18}$$

With increasing point numbers, we have

$$\operatorname{rank}(\boldsymbol{H}) \le \sum_{i=1}^{N} \operatorname{rank}\left[w_i \boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)\right] = N(n-1). \tag{19}$$

That is to say only a few pairs of points can hardly obtain accurate registration result of $\boldsymbol{R}$ and $\boldsymbol{t}$. This requirement is easy to be satisfied as the point numbers are usually far larger than the dimension so as to guarantee the global observability of parameters to be estimated.

Here, we define $\boldsymbol{P}_n(\boldsymbol{x}_i)$ as the $\boldsymbol{P}(\boldsymbol{x}_i)$ for n-dimensional registration problem. By examples we can see the contents of $\boldsymbol{P}_n(\boldsymbol{x}_i)$ with various dimensions in (2), shown at the bottom of the page. Then, we can summarize the generalized form for n-dimensional case, i.e.,

$$\boldsymbol{P}_n(\boldsymbol{x}_i) = \begin{pmatrix} {}_2^n\boldsymbol{x}_i^\top & & & \\ & {}_3^n\boldsymbol{x}_i^\top & & \\ -x_{i,1}\boldsymbol{I} & & \ddots & {}_{n-1}^n\boldsymbol{x}_i^\top \\ & -x_{i,2}\boldsymbol{I} & & \\ & & \ddots & -x_{i,n}\boldsymbol{I} \end{pmatrix} \tag{20}$$

where ${}_j^k\boldsymbol{x}_i$ denotes the partial column vector consisting of sequential components of $\boldsymbol{x}_i$ from index j up to index k. With the aid of explicit form of matrix such as those in (2), one can immediately infer that $\boldsymbol{P}_n^\top(\boldsymbol{x}_i)\boldsymbol{P}_n(\boldsymbol{x}_i)$ takes following form:

$$\boldsymbol{P}_n^\top(\boldsymbol{x}_i)\boldsymbol{P}_n(\boldsymbol{x}_i) = \begin{pmatrix} \boldsymbol{S}_1 & & & \iddots \\ & \boldsymbol{S}_2 & & \\ & & \ddots & \\ \iddots & & & \boldsymbol{S}_{n-2} \end{pmatrix} \tag{21}$$

in which $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_{n-2}$ are symmetric positive semidefinite matrices. Taking the fourth-order transformation as example, the matrix manipulation is shown in (14), shown at the bottom of the page. Moreover, for $\boldsymbol{S}_k$ we have the following eigenvalues:

$$\boldsymbol{P}_4^\top(\boldsymbol{x}_i)\boldsymbol{P}_4(\boldsymbol{x}_i) = \left(\begin{array}{ccc:ccc:c} \underbrace{\begin{matrix} x_{i,1}^2+x_{i,2}^2 & x_{i,2}x_{i,3} & x_{i,2}x_{i,4} \\ x_{i,2}x_{i,3} & x_{i,1}^2+x_{i,3}^2 & x_{i,3}x_{i,4} \\ x_{i,2}x_{i,4} & x_{i,3}x_{i,4} & x_{i,1}^2+x_{i,4}^2 \end{matrix}}_{\boldsymbol{S}_1} & & & & & & \iddots \\ & & & \underbrace{\begin{matrix} x_{i,2}^2+x_{i,3}^2 & x_{i,3}x_{i,4} \\ x_{i,3}x_{i,4} & x_{i,2}^2+x_{i,4}^2 \end{matrix}}_{\boldsymbol{S}_2} & & \iddots & \\ & & & & \ddots & & \\ & \iddots & & & & \underbrace{x_{i,3}^2+x_{i,4}^2}_{\boldsymbol{S}_3} & \end{array}\right) \tag{14}$$

$$\lambda_{S_k} = \underbrace{x_{i,k}^2, x_{i,k}^2, \ldots, x_{i,k}^2, \sum_{j=k}^{n} x_{i,j}^2}_{(n-k)\ \text{eigenvalues.}} \tag{22}$$

Then, $\boldsymbol{S}_k$ is positive semidefinite as all its eigenvalues are nonnegative. Therefore, the sum $\boldsymbol{H}$ is also symmetry-preserving and positive semidefinite and takes the summed form of (21). Invoking the Sherman–Morrison–Woodbury formula

$$(\boldsymbol{A}+\boldsymbol{BDC})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\Big(\boldsymbol{D}^{-1}+\boldsymbol{CA}^{-1}\boldsymbol{B}\Big)^{-1}\boldsymbol{CA}^{-1} \tag{23}$$

where $\boldsymbol{A}$, $\boldsymbol{D}$, and $\boldsymbol{D}^{-1}+\boldsymbol{CA}^{-1}\boldsymbol{B}$ are nonsingular matrices, we have

$$\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}^{-1} = \begin{bmatrix} \big(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\big)^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\big(\boldsymbol{D}-\boldsymbol{CA}^{-1}\boldsymbol{B}\big)^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\big(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\big)^{-1} & \big(\boldsymbol{D}-\boldsymbol{CA}^{-1}\boldsymbol{B}\big)^{-1}\end{bmatrix}.$$

Notice that

$$\Big(\boldsymbol{D}-\boldsymbol{CA}^{-1}\boldsymbol{B}\Big)^{-1} = \boldsymbol{D}^{-1}+\boldsymbol{D}^{-1}\boldsymbol{C}\Big(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\Big)^{-1}\boldsymbol{BD}^{-1} \tag{24}$$

we actually need to pay more attention to solving $\boldsymbol{A}^{-1}, \boldsymbol{D}^{-1}$, and $(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C})^{-1}$. Here, $\boldsymbol{A}^{-1}$ and $(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C})^{-1}$ have the same dimension and require the complexity of $O[(n-1)^3]$. Empirically, we found that all $\boldsymbol{S}_k$ matrices for $k = 1, 2, \ldots$ are strictly invertible when the problem (4) is valid, i.e., 1) the measurement number N is greater than the unknown degree $n(n-1)/2+n$ and 2) not all the points are coplanar or collinear. Then following the mapping of $\boldsymbol{H}$ to $\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{pmatrix}$, we can see that $\boldsymbol{D}$ here is also in the form of $\boldsymbol{H}$, i.e., the sum of multiple (21). In this way, $\boldsymbol{H}$ can be iteratively built up by smaller matrix blocks and the final computation comes from these blocks.

In numerical calculation of $\boldsymbol{H}^{-1}$, $\boldsymbol{H}$ may always suffer from very tiny or very large determinant values. At such time, the inverse of $\boldsymbol{H}$ will be influenced by the numerical loss. Using the above proposed fast blocked-matrix inversion, we may find out that $\boldsymbol{A}^{-1}$, $\boldsymbol{D}^{-1}$, and $(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C})^{-1}$ are required. While seen from (21), one can obviously notice that $\boldsymbol{A}$ and $\boldsymbol{D}$ are positive semidefinite. That is to say in engineering tasks, with sufficient point numbers, $\boldsymbol{A}, \boldsymbol{D}$ will be exactly positive definite and this solver is able to obtain much more robust blocked inversions. By taking matrix manipulations including addition and multiplication, $\boldsymbol{H}^{-1}$ can be accurately recovered. This shows the better robustness of this approach than conventional numerical schemes for computing $\boldsymbol{H}^{-1}$.

D. Singularity Avoidance

In (16), each independent component $\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)$ is singular due to the rank shown in (18). However, with the sum of heterogeneous matrices $\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)$, the matrix gradually reaches to full-rank state, which has been described in (19). When some extreme cases occur, $\boldsymbol{H}$ will be singular and the proposed fast inversion in the last section will also be trivial. The extreme case mainly covers.

1) Lack of mandatory point numbers [10].
2) Collinear reference vectors [43].
3) Dominant weights [44].

Each of these factors shown above will lead to numerical problem such that the matrix $\boldsymbol{H}$ is rank-deficient. In fact for almost all extreme cases, the determination of rotation would be meaningless as the observability of angles has been significantly distorted. Studying the extreme performance for the proposed method is to prevent the computation from numerical crash, e.g., the existence of not-a-number (NaN). For instance, when $\boldsymbol{H}$ is singular, the determinant will be very small which generates very large values of $\boldsymbol{H}^{-1}$ by

$$\boldsymbol{H}^{-1} = \mathrm{adj}(\boldsymbol{H})/\det(\boldsymbol{H}). \tag{25}$$

Then, inserting the large values into (4) results in another huge numerical loss. Under such circumstance, we should replace the original inversion with the pseudo inverse [45] $\boldsymbol{g} = \boldsymbol{H}^{+}\boldsymbol{v}$. Here, $\boldsymbol{H}^{+}$ is the Moore–Penrose generalized inverse of $\boldsymbol{H}$.

E. Covariance Analysis

There are some assumptions for us to derive the covariances of the obtained results in previous sections.

1) The measured points from one point set contain no correlated covariance between each other.
2) The two point sets for registration have independent noise distribution.

These assumptions are based on the fact that current Lidar sensor outputs huge loads of points, such as Velodyne VLP-16 Lidar can have 300 000 points per second. Thus, one can hardly describe individual correlation inside points. We use $\langle\cdot\rangle$ to represent the operation of expectation and δ is employed to represent an perturbed infinitesimal induced by input noises from $\boldsymbol{b}_i$ and $\boldsymbol{r}_i$. In this section, high-order (second-order or higher) infinitesimals are ignored for their tiny impacts in the error propagation. Then, the difference of $\boldsymbol{H}$ is

$$\begin{aligned}\delta\boldsymbol{H} &= \sum_{i=1}^{N} w_i\boldsymbol{P}^\top(\boldsymbol{x}_i+\delta\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i+\delta\boldsymbol{x}_i) - \sum_{i=1}^{N} w_i\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)\\ &= \sum_{i=1}^{N} w_i\begin{bmatrix}\boldsymbol{P}^\top(\delta\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)+\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\delta\boldsymbol{x}_i)\\ +\boldsymbol{P}^\top(\delta\boldsymbol{x}_i)\boldsymbol{P}(\delta\boldsymbol{x}_i)\end{bmatrix}\\ &\approx \sum_{i=1}^{N} w_i\Big[\boldsymbol{P}^\top(\delta\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)+\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\delta\boldsymbol{x}_i)\Big].\end{aligned} \tag{26}$$

Note that for each subitem $\boldsymbol{J}_i$, it follows that:

$$\boldsymbol{J}_i = w_i\Big[\boldsymbol{P}^\top(\delta\boldsymbol{x}_i)\boldsymbol{P}(\boldsymbol{x}_i)+\boldsymbol{P}^\top(\boldsymbol{x}_i)\boldsymbol{P}(\delta\boldsymbol{x}_i)\Big]. \tag{27}$$

It can be decomposed linearly in $\delta\boldsymbol{x}_i$, such that

$$\boldsymbol{J}_i = \Big[\boldsymbol{K}_1(\boldsymbol{x}_i)\delta\boldsymbol{x}_i, \boldsymbol{K}_2(\boldsymbol{x}_i)\delta\boldsymbol{x}_i, \ldots, \boldsymbol{K}_{\frac{n(n-1)}{2}}(\boldsymbol{x}_i)\delta\boldsymbol{x}_i\Big]. \tag{28}$$

For instance when $n = 4$, the matrices are

$$\boldsymbol{K}_1(\boldsymbol{x}_i) = w_i\begin{pmatrix}2x_{i,1} & 2x_{i,2} & 0 & 0\\ 0 & x_{i,3} & x_{i,2} & 0\\ 0 & x_{i,4} & 0 & x_{i,2}\\ -x_{i,3} & 0 & -x_{i,1} & 0\\ -x_{i,4} & 0 & 0 & -x_{i,1}\\ 0 & 0 & 0 & 0\end{pmatrix}$$

$$
\begin{aligned}
\boldsymbol{K}_2(\boldsymbol{x}_i) &= w_i \begin{pmatrix} 0 & x_{i,3} & x_{i,2} & 0 \\ 2x_{i,1} & 0 & 2x_{i,3} & 0 \\ 0 & 0 & x_{i,4} & x_{i,3} \\ x_{i,2} & x_{i,1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -x_{i,4} & 0 & 0 & -x_{i,1} \end{pmatrix} \\
\boldsymbol{K}_3(\boldsymbol{x}_i) &= w_i \begin{pmatrix} 0 & x_{i,4} & 0 & x_{i,2} \\ 0 & 0 & x_{i,4} & x_{i,3} \\ 2x_{i,1} & 0 & 0 & 2x_{i,4} \\ 0 & 0 & 0 & 0 \\ x_{i,2} & x_{i,1} & 0 & 0 \\ x_{i,3} & 0 & x_{i,1} & 0 \end{pmatrix} \\
\boldsymbol{K}_4(\boldsymbol{x}_i) &= w_i \begin{pmatrix} -x_{i,3} & 0 & -x_{i,1} & 0 \\ x_{i,2} & x_{i,1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 2x_{i,2} & 2x_{i,3} & 0 \\ 0 & 0 & x_{i,4} & x_{i,3} \\ 0 & -x_{i,4} & 0 & -x_{i,2} \end{pmatrix} \\
\boldsymbol{K}_5(\boldsymbol{x}_i) &= w_i \begin{pmatrix} -x_{i,4} & 0 & 0 & -x_{i,1} \\ 0 & 0 & 0 & 0 \\ x_{i,2} & x_{i,1} & 0 & 0 \\ 0 & 0 & x_{i,4} & x_{i,3} \\ 0 & 2x_{i,2} & 0 & 2x_{i,4} \\ 0 & x_{i,3} & x_{i,2} & 0 \end{pmatrix} \\
\boldsymbol{K}_6(\boldsymbol{x}_i) &= w_i \begin{pmatrix} 0 & 0 & 0 & 0 \\ -x_{i,4} & 0 & 0 & -x_{i,1} \\ x_{i,3} & 0 & x_{i,1} & 0 \\ 0 & -x_{i,4} & 0 & -x_{i,2} \\ 0 & x_{i,3} & x_{i,2} & 0 \\ 0 & 0 & 2x_{i,3} & 2x_{i,4} \end{pmatrix}.
\end{aligned} \tag{29}
$$

The noise-perturbed system is given as follows:

$$
\begin{aligned}
&(\boldsymbol{H} + \delta\boldsymbol{H})(\boldsymbol{g} + \delta\boldsymbol{g}) = \boldsymbol{v} + \delta\boldsymbol{v} \\
&\Rightarrow \delta\boldsymbol{H}\boldsymbol{g} + (\boldsymbol{H} + \delta\boldsymbol{H})\delta\boldsymbol{g} = \delta\boldsymbol{v} \\
&\Rightarrow \delta\boldsymbol{g} = \boldsymbol{H}^{+}(\delta\boldsymbol{v} - \delta\boldsymbol{H}\boldsymbol{g})
\end{aligned} \tag{30}
$$

where

$$
\begin{aligned}
\delta\boldsymbol{H}\boldsymbol{g} &= \sum_{i=1}^{N} \boldsymbol{J}_i \boldsymbol{g} \\
&= \sum_{i=1}^{N} \sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j(\boldsymbol{x}_i)\delta\boldsymbol{x}_i = \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j(\boldsymbol{x}_i) \right] \delta\boldsymbol{x}_i.
\end{aligned} \tag{31}
$$

We obtain

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\boldsymbol{g}} &= \left\langle \delta\boldsymbol{g}\delta\boldsymbol{g}^{\top} \right\rangle \\
&= \left\langle \boldsymbol{H}^{+}(\delta\boldsymbol{v} - \delta\boldsymbol{H}\boldsymbol{g})(\delta\boldsymbol{v} - \delta\boldsymbol{H}\boldsymbol{g})^{\top}\boldsymbol{H}^{+} \right\rangle \\
&= \left\langle \boldsymbol{H}^{+} \begin{pmatrix} \delta\boldsymbol{v}\delta\boldsymbol{v}^{\top} + \delta\boldsymbol{H}\boldsymbol{g}\boldsymbol{g}^{\top}\delta\boldsymbol{H} \\ -\delta\boldsymbol{H}\boldsymbol{g}\delta\boldsymbol{v}^{\top} - \delta\boldsymbol{v}\boldsymbol{g}^{\top}\delta\boldsymbol{H} \end{pmatrix} \boldsymbol{H}^{+} \right\rangle \\
&= \boldsymbol{H}^{+} \begin{pmatrix} \boldsymbol{\Sigma}_{\boldsymbol{v}} \\ + \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j(\boldsymbol{x}_i) \right] \boldsymbol{\Sigma}_{\boldsymbol{x}_i} \\ \times \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j^{\top}(\boldsymbol{x}_i) \right] \\ - \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j(\boldsymbol{x}_i) \right] \left\langle \delta\boldsymbol{x}_i\delta\boldsymbol{v}^{\top} \right\rangle \\ -\left\langle \delta\boldsymbol{x}_i\delta\boldsymbol{v}^{\top} \right\rangle \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j^{\top}(\boldsymbol{x}_i) \right] \end{pmatrix} \boldsymbol{H}^{+}
\end{aligned} \tag{32}
$$

in which

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\boldsymbol{v}} &= \left\langle \delta\boldsymbol{v}\delta\boldsymbol{v}^{\top} \right\rangle \\
&\approx \sum_{i=1}^{N} w_i^2 \left\langle \begin{matrix} \left\{ \boldsymbol{P}^{\top}(\delta\boldsymbol{x}_i)\boldsymbol{d}_i + \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\delta\boldsymbol{d}_i \right\} \\ \left\{ \boldsymbol{P}^{\top}(\delta\boldsymbol{x}_i)\boldsymbol{d}_i + \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\delta\boldsymbol{d}_i \right\}^{\top} \end{matrix} \right\rangle \\
&= \sum_{i=1}^{N} w_i^2 \begin{bmatrix} \boldsymbol{P}^{\top}(\boldsymbol{d}_i)\boldsymbol{\Sigma}_{\boldsymbol{x}_i}\boldsymbol{P}(\boldsymbol{d}_i) - \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{d}_i}\boldsymbol{P}(\boldsymbol{d}_i) \\ -\boldsymbol{P}^{\top}(\boldsymbol{d}_i)\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{d}_i}\boldsymbol{P}(\boldsymbol{x}_i) + \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_{\boldsymbol{d}_i}\boldsymbol{P}(\boldsymbol{x}_i) \end{bmatrix}
\end{aligned} \tag{33}
$$

and

$$
\begin{aligned}
\left\langle \delta\boldsymbol{x}_i\delta\boldsymbol{v}^{\top} \right\rangle &= \left\langle \delta\boldsymbol{x}_i \sum_{i=1}^{N} w_i \left[\boldsymbol{P}^{\top}(\delta\boldsymbol{x}_i)\boldsymbol{d}_i + \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\delta\boldsymbol{d}_i \right]^{\top} \right\rangle \\
&= \left\langle \sum_{i=1}^{N} w_i \left[\delta\boldsymbol{x}_i\boldsymbol{d}_i^{\top}\boldsymbol{P}(\delta\boldsymbol{x}_i) + \delta\boldsymbol{x}_i\delta\boldsymbol{d}_i^{\top}\boldsymbol{P}(\boldsymbol{x}_i) \right] \right\rangle \\
&= \left\langle \sum_{i=1}^{N} w_i \left[\delta\boldsymbol{x}_i\delta\boldsymbol{d}_i^{\top}\boldsymbol{P}(\boldsymbol{x}_i) - \delta\boldsymbol{x}_i\delta\boldsymbol{x}_i^{\top}\boldsymbol{P}(\boldsymbol{d}_i) \right] \right\rangle \\
&= \sum_{i=1}^{N} w_i \left[\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{d}_i}\boldsymbol{P}(\boldsymbol{x}_i) - \boldsymbol{\Sigma}_{\boldsymbol{x}_i}\boldsymbol{P}(\boldsymbol{d}_i) \right]
\end{aligned} \tag{34}
$$

provided that

$$
\delta\boldsymbol{v} = \sum_{i=1}^{N} w_i \left[\boldsymbol{P}^{\top}(\delta\boldsymbol{x}_i)\boldsymbol{d}_i + \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\delta\boldsymbol{d}_i \right] \tag{35}
$$

and

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\boldsymbol{x}_i,\boldsymbol{d}_i} &= \left\langle \delta\boldsymbol{x}_i\delta\boldsymbol{d}_i^{\top} \right\rangle \\
&= \left\langle \left(\delta\tilde{\boldsymbol{b}}_i + \delta\tilde{\boldsymbol{r}}_i \right) \left(\delta\tilde{\boldsymbol{r}}_i - \delta\tilde{\boldsymbol{b}}_i \right)^{\top} \right\rangle \\
&= \left\langle \delta\tilde{\boldsymbol{b}}_i\delta\tilde{\boldsymbol{r}}_i^{\top} + \delta\tilde{\boldsymbol{r}}_i\delta\tilde{\boldsymbol{r}}_i^{\top} - \delta\tilde{\boldsymbol{b}}_i\delta\tilde{\boldsymbol{b}}_i^{\top} - \delta\tilde{\boldsymbol{r}}_i\delta\tilde{\boldsymbol{b}}_i^{\top} \right\rangle \\
&= \left\langle \delta\tilde{\boldsymbol{b}}_i\delta\tilde{\boldsymbol{r}}_i^{\top} + \delta\tilde{\boldsymbol{r}}_i\delta\tilde{\boldsymbol{r}}_i^{\top} - \delta\tilde{\boldsymbol{b}}_i\delta\tilde{\boldsymbol{b}}_i^{\top} - \delta\tilde{\boldsymbol{r}}_i\delta\tilde{\boldsymbol{b}}_i^{\top} \right\rangle \\
&= \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}_i} - \boldsymbol{\Sigma}_{\tilde{\boldsymbol{b}}_i}.
\end{aligned} \tag{36}
$$

In particular, when $\tilde{\boldsymbol{b}}_i$ and $\tilde{\boldsymbol{r}}_i$ have the same statistical distribution, especially in the cases with huge numbers of points, the covariance of $\boldsymbol{g}$ is

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\boldsymbol{g}} &= \boldsymbol{H}^{+} \\
&\times \begin{pmatrix} \sum_{i=1}^{N} w_i \left[\boldsymbol{P}^{\top}(\boldsymbol{d}_i)\boldsymbol{\Sigma}_{\boldsymbol{x}_i}\boldsymbol{P}(\boldsymbol{d}_i) + \boldsymbol{P}^{\top}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_{\boldsymbol{d}_i}\boldsymbol{P}(\boldsymbol{x}_i) \right] \\ + \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j(\boldsymbol{x}_i) \right] \boldsymbol{\Sigma}_{\boldsymbol{x}_i} \sum_{i=1}^{N} \left[\sum_{j=1}^{\frac{n(n-1)}{2}} g_j \boldsymbol{K}_j^{\top}(\boldsymbol{x}_i) \right] \end{pmatrix} \boldsymbol{H}^{+}.
\end{aligned} \tag{37}
$$

Algorithm 1 Generalized Linear n-Dimensional Registration

Require: Point clouds with N pairs of measurements $\boldsymbol{r}_i$ and $\boldsymbol{b}_i$ for $i = 1, 2, \ldots, N$.
Step 1: Compute mean points $\bar{\boldsymbol{b}} = \sum_{i=1}^{n} a_i \boldsymbol{b}_i, \bar{\boldsymbol{r}} = \sum_{i=1}^{n} a_i \boldsymbol{r}_i$.
Step 2: Compute $\boldsymbol{H}$ matrix and $\boldsymbol{v}$ vector using (21). If N is large, use results in (25) $\sim$ (30) to simplify the process.
Step 3: Compute required elements of $\boldsymbol{g}$ in (20) and then compute the rotation using (22).
Step 4: Reconstruct the translation by $\boldsymbol{t} = \bar{\boldsymbol{b}} - \boldsymbol{C}\bar{\boldsymbol{r}}$.
Step 5: Compute the covariances of $\boldsymbol{R}$ and $\boldsymbol{t}$ by (46) and (47) respectively.

While in applications we always need uncertainty description of $\boldsymbol{R}$. In this way, one may derive

$$\begin{cases} (\boldsymbol{I}+\boldsymbol{G})\boldsymbol{R} = \boldsymbol{I}-\boldsymbol{G} \\ \big[\boldsymbol{I}+\boldsymbol{G}(\boldsymbol{g}+\delta\boldsymbol{g})\big](\boldsymbol{R}+\delta\boldsymbol{R}) = \boldsymbol{I}-\boldsymbol{G}(\boldsymbol{g}+\delta\boldsymbol{g}) \\ \quad \Rightarrow (\boldsymbol{R}+\delta\boldsymbol{R}) + \boldsymbol{G}(\boldsymbol{g}+\delta\boldsymbol{g})(\boldsymbol{R}+\delta\boldsymbol{R}) = \boldsymbol{I}-\boldsymbol{G}(\boldsymbol{g}+\delta\boldsymbol{g}) \\ \quad \Rightarrow \delta\boldsymbol{R} + \boldsymbol{G}\delta\boldsymbol{R} + \boldsymbol{G}(\delta\boldsymbol{g})\boldsymbol{R} = -\boldsymbol{G}(\delta\boldsymbol{g}) \\ \quad \Rightarrow \delta\boldsymbol{R} = -(\boldsymbol{I}+\boldsymbol{G})^{-1}\delta\boldsymbol{g}_{\otimes}(\boldsymbol{R}+\boldsymbol{I}). \end{cases} \tag{38}$$

Let us write $\boldsymbol{R}+\boldsymbol{I}$ in columns as $\boldsymbol{R}+\boldsymbol{I} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n)$. Then, one has

$$\delta\boldsymbol{g}_{\otimes}(\boldsymbol{R}+\boldsymbol{I}) = \big[\boldsymbol{P}(\boldsymbol{s}_1)\delta\boldsymbol{g}, \boldsymbol{P}(\boldsymbol{s}_2)\delta\boldsymbol{g}, \ldots, \boldsymbol{P}(\boldsymbol{s}_n)\delta\boldsymbol{g}\big]. \tag{39}$$

Finally, the covariance of $\boldsymbol{R}$ is

$$\begin{aligned} \boldsymbol{\Sigma}_{\boldsymbol{R}} &= \left\langle \delta\boldsymbol{R}\delta\boldsymbol{R}^{\top} \right\rangle \\ &= \left\langle \begin{matrix} (\boldsymbol{I}+\boldsymbol{G})^{-1}\delta\boldsymbol{g}_{\otimes}(\boldsymbol{R}+\boldsymbol{I}) \\ (\boldsymbol{R}+\boldsymbol{I})^{\top}\delta\boldsymbol{g}_{\otimes}\big[(\boldsymbol{I}+\boldsymbol{G})^{-1}\big]^{\top} \end{matrix} \right\rangle \\ &= \left\langle \begin{matrix} (\boldsymbol{I}+\boldsymbol{G})^{-1} \\ \big[\sum_{k=1}^{n} \boldsymbol{P}(\boldsymbol{s}_k)\delta\boldsymbol{g}\delta\boldsymbol{g}^{\top}\boldsymbol{P}^{\top}(\boldsymbol{s}_k)\big]\big[(\boldsymbol{I}+\boldsymbol{G})^{-1}\big]^{\top} \end{matrix} \right\rangle \\ &= (\boldsymbol{I}+\boldsymbol{G})^{-1}\left[\sum_{k=1}^{n} \boldsymbol{P}(\boldsymbol{s}_k)\boldsymbol{\Sigma}_{\boldsymbol{g}}\boldsymbol{P}^{\top}(\boldsymbol{s}_k)\right]\Big[(\boldsymbol{I}+\boldsymbol{G})^{-1}\Big]^{\top}. \end{aligned} \tag{40}$$

Likewise, the error model of $\boldsymbol{t}$ is $\delta\boldsymbol{t} = \delta\boldsymbol{b} - \delta\boldsymbol{R}\bar{\boldsymbol{r}} - \boldsymbol{R}\delta\boldsymbol{r}$, which leads to

$$\begin{aligned} \boldsymbol{\Sigma}_{\boldsymbol{t}} &\approx \boldsymbol{\Sigma}_{\boldsymbol{b}} + \boldsymbol{R}\boldsymbol{\Sigma}_{\boldsymbol{r}}\boldsymbol{R}^{\top} \\ &\quad + \left\langle \begin{matrix} (\boldsymbol{I}+\boldsymbol{G})^{-1}\delta\boldsymbol{g}_{\otimes}(\boldsymbol{R}+\boldsymbol{I})\bar{\boldsymbol{r}} \\ \bar{\boldsymbol{r}}^{\top}(\boldsymbol{R}+\boldsymbol{I})^{\top}\delta\boldsymbol{g}_{\otimes}\big[(\boldsymbol{I}+\boldsymbol{G})^{-1}\big]^{\top} \end{matrix} \right\rangle \\ &\approx \boldsymbol{\Sigma}_{\boldsymbol{b}} + \boldsymbol{R}\boldsymbol{\Sigma}_{\boldsymbol{r}}\boldsymbol{R}^{\top} \\ &\quad + (\boldsymbol{I}+\boldsymbol{G})^{-1}\left[\sum_{k=1}^{n} \boldsymbol{P}(\bar{r}_k\boldsymbol{s}_k)\boldsymbol{\Sigma}_{\boldsymbol{g}}\boldsymbol{P}^{\top}(\bar{r}_k\boldsymbol{s}_k)\right]\Big[(\boldsymbol{I}+\boldsymbol{G})^{-1}\Big]^{\top}. \end{aligned} \tag{41}$$

III. Experimental Results

A. Accuracy and Robustness

The algorithmic flow is show in Algorithm 1. For the Sections III-A and III-B, we simulate the following vector pairs:

$$\boldsymbol{b}_i = \boldsymbol{R}_{\text{true}}\boldsymbol{r}_i + \boldsymbol{t}_{\text{true}} + \boldsymbol{\varepsilon}, \quad i = 1, 2, \ldots, N \tag{42}$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \varepsilon\boldsymbol{I})$ is the zero-mean random perturbation subject to the Gaussian distribution and $\boldsymbol{R}_{\text{true}}, \boldsymbol{t}_{\text{true}}$ are true reference rotation matrix and translation vector. ε here is called the noise density that decides the magnitude of the noise. The accuracy of the n-dimensional registration will be affected by the value of ε, i.e., with larger ε the estimation accuracy will be worse. Six cases with different values of ε are simulated uniformly where the point number $N = 100\,000$ to guarantee the average statistical performance. The following error criterion is invoked:

$$\epsilon = \frac{1}{N}\sum_{i=1}^{N} \big\|\boldsymbol{b}_i - \boldsymbol{R}\boldsymbol{r}_i - \boldsymbol{t}\big\|^2. \tag{43}$$

We pick up the SVD [46] and LMI [28] approaches for comparison. The SVD uses the following principle:

$$\boldsymbol{R} = \boldsymbol{U}\ \text{diag}[1, 1, \ldots, \det(\boldsymbol{U}\boldsymbol{V})]\boldsymbol{V}^{\top} \tag{44}$$

with SVD of $\boldsymbol{Z}$ such that

$$\begin{aligned} \boldsymbol{U}\boldsymbol{\Pi}\boldsymbol{V}^{\top} &= \boldsymbol{Z} \\ \boldsymbol{Z} &= \frac{1}{N}\sum_{i=1}^{N}\Big(\boldsymbol{b}_i - \bar{\boldsymbol{b}}\Big)(\boldsymbol{r}_i - \bar{\boldsymbol{r}})^{\top} \end{aligned} \tag{45}$$

where $\boldsymbol{\Pi}$ is the diagonal matrix containing all singular values of $\boldsymbol{Z}$. The LMI is based on the following optimization:

$$\underset{\boldsymbol{R}\in SO(n)}{\arg\max}\ \ \text{tr}(\boldsymbol{R}\boldsymbol{Z}^{\top}) \tag{46}$$

subject to

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{R}^{\top} \\ \boldsymbol{R} & \boldsymbol{I} \end{pmatrix} \geq \boldsymbol{0}. \tag{47}$$

All these algorithms are implemented using the C++ programming language of the standard 2014 and compiled with the GNU gcc 7.0 compiler. We use the Eigen library to compute the SVD and related matrix manipulations. The `GpoSolver` is adopted for the solution of LMI [47]. Each algorithm is executed for 100 times for mean errors. Simulations on different dimensions are also reflected in the presented tables. The results are shown in Tables I–VI.

One can see that with low values of ε, all the algorithms are able to generate accurate registration results. Among them, the SVD owns the best precision and the proposed method ranks the second. With growing ε, the noise also grows, inducing higher errors to the model (42). The proposed algorithm now has the same accuracy with that of SVD while the LMI gradually performs badly. The reason is that SVD and the proposed algorithm share the same optimization framework and all belong to analytical solutions. However, LMI converts the rigid-registration loss function into a polynomial optimization one that requires a sophisticated nonlinear solver. As SVD has been regarded as a highly reliable algorithm [48], [49], that is to say the proposed method is stable and reliable in this sense.

TABLE I
ACCURACY (NOISE DENSITY $\varepsilon = 1.0 \times 10^{-15}$)

Dimension n	ϵ_{SVD}	ϵ_{LMI}	$\epsilon_{\text{Proposed}}$
5	2.5189×10^{-15}	3.9770×10^{-15}	$\mathbf{2.5534 \times 10^{-15}}$
10	4.1445×10^{-15}	7.5453×10^{-15}	$\mathbf{4.1869 \times 10^{-15}}$
15	6.1569×10^{-15}	9.8278×10^{-15}	$\mathbf{6.1933 \times 10^{-15}}$
20	7.4839×10^{-15}	1.2049×10^{-14}	$\mathbf{7.5265 \times 10^{-15}}$
25	9.4027×10^{-15}	1.4780×10^{-14}	$\mathbf{9.4433 \times 10^{-15}}$
30	8.9694×10^{-15}	1.3265×10^{-14}	$\mathbf{9.0142 \times 10^{-15}}$
35	1.0416×10^{-14}	1.5201×10^{-14}	$\mathbf{1.0466 \times 10^{-14}}$
40	1.1447×10^{-14}	1.7480×10^{-14}	$\mathbf{1.1496 \times 10^{-14}}$
45	1.2971×10^{-14}	2.0592×10^{-14}	$\mathbf{1.3020 \times 10^{-14}}$
50	1.4288×10^{-14}	2.2007×10^{-14}	$\mathbf{1.4339 \times 10^{-14}}$
55	1.3608×10^{-14}	2.2283×10^{-14}	$\mathbf{1.3661 \times 10^{-14}}$
60	1.4685×10^{-14}	2.7381×10^{-14}	$\mathbf{1.4741 \times 10^{-14}}$
65	1.5838×10^{-14}	2.5491×10^{-14}	$\mathbf{1.5894 \times 10^{-14}}$
70	1.7252×10^{-14}	2.7601×10^{-14}	$\mathbf{1.7306 \times 10^{-14}}$
75	1.7786×10^{-14}	3.0958×10^{-14}	$\mathbf{1.7842 \times 10^{-14}}$
80	1.9007×10^{-14}	3.2302×10^{-14}	$\mathbf{1.9062 \times 10^{-14}}$
85	1.9726×10^{-14}	3.1926×10^{-14}	$\mathbf{1.9782 \times 10^{-14}}$
90	2.1036×10^{-14}	3.4314×10^{-14}	$\mathbf{2.1094 \times 10^{-14}}$
95	2.0890×10^{-14}	2.9565×10^{-14}	$\mathbf{2.0948 \times 10^{-14}}$
100	2.2686×10^{-14}	3.7892×10^{-14}	$\mathbf{2.2732 \times 10^{-14}}$

TABLE II
ACCURACY (NOISE DENSITY $\varepsilon = 1.0 \times 10^{-11}$)

Dimension n	ϵ_{SVD}	ϵ_{LMI}	$\epsilon_{\text{Proposed}}$
5	8.0732×10^{-12}	1.3465×10^{-11}	$\mathbf{8.0732 \times 10^{-12}}$
10	1.2590×10^{-11}	1.6765×10^{-11}	$\mathbf{1.2590 \times 10^{-11}}$
15	1.5558×10^{-11}	2.4746×10^{-11}	$\mathbf{1.5558 \times 10^{-11}}$
20	1.7976×10^{-11}	3.0183×10^{-11}	$\mathbf{1.7976 \times 10^{-11}}$
25	2.0297×10^{-11}	3.4387×10^{-11}	$\mathbf{2.0297 \times 10^{-11}}$
30	2.2206×10^{-11}	3.8058×10^{-11}	$\mathbf{2.2206 \times 10^{-11}}$
35	2.4105×10^{-11}	3.1602×10^{-11}	$\mathbf{2.4105 \times 10^{-11}}$
40	2.5741×10^{-11}	3.8339×10^{-11}	$\mathbf{2.5741 \times 10^{-11}}$
45	2.7409×10^{-11}	4.2927×10^{-11}	$\mathbf{2.7409 \times 10^{-11}}$
50	2.8842×10^{-11}	4.9149×10^{-11}	$\mathbf{2.8842 \times 10^{-11}}$
55	3.0359×10^{-11}	4.5121×10^{-11}	$\mathbf{3.0359 \times 10^{-11}}$
60	3.1765×10^{-11}	5.1417×10^{-11}	$\mathbf{3.1765 \times 10^{-11}}$
65	3.3025×10^{-11}	5.3235×10^{-11}	$\mathbf{3.3025 \times 10^{-11}}$
70	3.4349×10^{-11}	5.5118×10^{-11}	$\mathbf{3.4349 \times 10^{-11}}$
75	3.5478×10^{-11}	5.5461×10^{-11}	$\mathbf{3.5478 \times 10^{-11}}$
80	3.6685×10^{-11}	5.7690×10^{-11}	$\mathbf{3.6685 \times 10^{-11}}$
85	3.7946×10^{-11}	5.9655×10^{-11}	$\mathbf{3.7946 \times 10^{-11}}$
90	3.9061×10^{-11}	6.8401×10^{-11}	$\mathbf{3.9061 \times 10^{-11}}$
95	4.0109×10^{-11}	6.7446×10^{-11}	$\mathbf{4.0109 \times 10^{-11}}$
100	4.0974×10^{-11}	7.6927×10^{-11}	$\mathbf{4.0974 \times 10^{-11}}$

TABLE III
ACCURACY (NOISE DENSITY $\varepsilon = 1.0 \times 10^{-08}$)

Dimension n	ϵ_{SVD}	ϵ_{LMI}	$\epsilon_{\text{Proposed}}$
5	7.7914×10^{-09}	1.1886×10^{-08}	$\mathbf{7.7914 \times 10^{-09}}$
10	1.2518×10^{-08}	2.0765×10^{-08}	$\mathbf{1.2518 \times 10^{-08}}$
15	1.5479×10^{-08}	2.6433×10^{-08}	$\mathbf{1.5479 \times 10^{-08}}$
20	1.8111×10^{-08}	3.0359×10^{-08}	$\mathbf{1.8111 \times 10^{-08}}$
25	2.0241×10^{-08}	3.4652×10^{-08}	$\mathbf{2.0241 \times 10^{-08}}$
30	2.2276×10^{-08}	3.5577×10^{-08}	$\mathbf{2.2276 \times 10^{-08}}$
35	2.4160×10^{-08}	4.3726×10^{-08}	$\mathbf{2.4160 \times 10^{-08}}$
40	2.5811×10^{-08}	3.8046×10^{-08}	$\mathbf{2.5811 \times 10^{-08}}$
45	2.7455×10^{-08}	4.2292×10^{-08}	$\mathbf{2.7455 \times 10^{-08}}$
50	2.8837×10^{-08}	4.9336×10^{-08}	$\mathbf{2.8837 \times 10^{-08}}$
55	3.0398×10^{-08}	4.7447×10^{-08}	$\mathbf{3.0398 \times 10^{-08}}$
60	3.1659×10^{-08}	4.8594×10^{-08}	$\mathbf{3.1659 \times 10^{-08}}$
65	3.3004×10^{-08}	5.0344×10^{-08}	$\mathbf{3.3004 \times 10^{-08}}$
70	3.4303×10^{-08}	5.8933×10^{-08}	$\mathbf{3.4303 \times 10^{-08}}$
75	3.5634×10^{-08}	5.1539×10^{-08}	$\mathbf{3.5634 \times 10^{-08}}$
80	3.6726×10^{-08}	5.6076×10^{-08}	$\mathbf{3.6726 \times 10^{-08}}$
85	3.7855×10^{-08}	6.3566×10^{-08}	$\mathbf{3.7855 \times 10^{-08}}$
90	3.8955×10^{-08}	6.1485×10^{-08}	$\mathbf{3.8955 \times 10^{-08}}$
95	4.0017×10^{-08}	6.3347×10^{-08}	$\mathbf{4.0017 \times 10^{-08}}$
100	4.1120×10^{-08}	6.5319×10^{-08}	$\mathbf{4.1120 \times 10^{-08}}$

TABLE IV
ACCURACY (NOISE DENSITY $\varepsilon = 1.0 \times 10^{-05}$)

Dimension n	ϵ_{SVD}	ϵ_{LMI}	$\epsilon_{\text{Proposed}}$
5	8.0834×10^{-06}	1.2991×10^{-05}	$\mathbf{8.0834 \times 10^{-06}}$
10	1.2327×10^{-05}	1.9442×10^{-05}	$\mathbf{1.2327 \times 10^{-05}}$
15	1.5622×10^{-05}	2.7248×10^{-05}	$\mathbf{1.5622 \times 10^{-05}}$
20	1.7870×10^{-05}	2.8758×10^{-05}	$\mathbf{1.7870 \times 10^{-05}}$
25	2.0152×10^{-05}	3.0017×10^{-05}	$\mathbf{2.0152 \times 10^{-05}}$
30	2.2217×10^{-05}	3.6156×10^{-05}	$\mathbf{2.2217 \times 10^{-05}}$
35	2.4295×10^{-05}	3.9725×10^{-05}	$\mathbf{2.4295 \times 10^{-05}}$
40	2.5801×10^{-05}	3.5996×10^{-05}	$\mathbf{2.5801 \times 10^{-05}}$
45	2.7411×10^{-05}	4.2853×10^{-05}	$\mathbf{2.7411 \times 10^{-05}}$
50	2.8786×10^{-05}	4.6512×10^{-05}	$\mathbf{2.8786 \times 10^{-05}}$
55	3.0412×10^{-05}	4.4348×10^{-05}	$\mathbf{3.0412 \times 10^{-05}}$
60	3.1668×10^{-05}	4.7548×10^{-05}	$\mathbf{3.1668 \times 10^{-05}}$
65	3.3071×10^{-05}	5.0175×10^{-05}	$\mathbf{3.3071 \times 10^{-05}}$
70	3.4427×10^{-05}	4.9038×10^{-05}	$\mathbf{3.4427 \times 10^{-05}}$
75	3.5611×10^{-05}	6.3040×10^{-05}	$\mathbf{3.5611 \times 10^{-05}}$
80	3.6670×10^{-05}	5.9718×10^{-05}	$\mathbf{3.6670 \times 10^{-05}}$
85	3.7910×10^{-05}	6.1752×10^{-05}	$\mathbf{3.7910 \times 10^{-05}}$
90	3.8971×10^{-05}	6.7341×10^{-05}	$\mathbf{3.8971 \times 10^{-05}}$
95	4.0016×10^{-05}	6.3221×10^{-05}	$\mathbf{4.0016 \times 10^{-05}}$
100	4.1141×10^{-05}	6.9154×10^{-05}	$\mathbf{4.1141 \times 10^{-05}}$

B. Computational Efficiency

There is a slight impact of the point numbers on the computational efficiency since the complexity of the algorithm is linear with respect to the point numbers, as shown in (15). For high-order matrices, the proposed matrix inversion will gradually be slower than SVD. This is because the proposed one has the complexity of $O[(1/2)n(n-1)]^3$ while that of the SVD is $O(n^3)$. Choosing the slowest algorithm as a basic reference, the realistic time complexity of the other algorithms should be compentated for specific machines. This leads to the modified time complexity expressions for the proposed method and SVD i.e., $O(a_i n^3)$ with a_i the compensation constant for ith algorithm. For instance, if we implement the matrix operations with the famous Eigen library using C++ programming language on a MacBook Pro 2017 13" with CPU of i7-4core 3.5 GHz, the measured mean values (100 times) are $a_2/a_1 = 1.667\times10^5$ and $a_3/a_1 = 6.25\times10^6$. Then, the plots of time complexity can be shown in Fig. 1. The unaltered matrix inversion using Eigen is very slow in high dimensions and has very large slopes as the dimension n increases. It should be noted that a_i is determined by the machine and employed library. For lower dimensions, the computation time would be much lower than the SVD as low-dimensional matrix inversions can usually be obtained with algebraic results instantly. The intersection point between the proposed method and SVD is $n = 112$. A detailed comparison showing the exact in-run performance of the computation time is presented in Table VII where t denotes the evaluated computation time.

We may see that with increasing dimension n, the computational complexity also increases. At the initial stage, the SVD is quite slower than the proposed method and the LMI is always the slowest since it needs nonlinear optimizations.

TABLE V
ACCURACY NOISE DENSITY $\varepsilon = 1.0 \times 10^{-02}$

Dimension n	ϵ_{SVD}	ϵ_{LMI}	$\epsilon_{\text{Proposed}}$
5	8.1853×10^{-03}	1.4259×10^{-02}	$\mathbf{8.1853 \times 10^{-03}}$
10	1.2534×10^{-02}	1.9991×10^{-02}	$\mathbf{1.2534 \times 10^{-02}}$
15	1.5371×10^{-02}	2.3495×10^{-02}	$\mathbf{1.5371 \times 10^{-02}}$
20	1.8140×10^{-02}	2.6999×10^{-02}	$\mathbf{1.8140 \times 10^{-02}}$
25	2.0178×10^{-02}	3.0602×10^{-02}	$\mathbf{2.0178 \times 10^{-02}}$
30	2.2231×10^{-02}	3.7298×10^{-02}	$\mathbf{2.2231 \times 10^{-02}}$
35	2.4116×10^{-02}	3.9932×10^{-02}	$\mathbf{2.4116 \times 10^{-02}}$
40	2.5846×10^{-02}	3.9382×10^{-02}	$\mathbf{2.5846 \times 10^{-02}}$
45	2.7373×10^{-02}	3.8117×10^{-02}	$\mathbf{2.7373 \times 10^{-02}}$
50	2.8909×10^{-02}	4.3503×10^{-02}	$\mathbf{2.8909 \times 10^{-02}}$
55	3.0338×10^{-02}	5.0324×10^{-02}	$\mathbf{3.0338 \times 10^{-02}}$
60	3.1645×10^{-02}	4.9776×10^{-02}	$\mathbf{3.1645 \times 10^{-02}}$
65	3.3087×10^{-02}	5.1069×10^{-02}	$\mathbf{3.3087 \times 10^{-02}}$
70	3.4326×10^{-02}	5.6569×10^{-02}	$\mathbf{3.4326 \times 10^{-02}}$
75	3.5564×10^{-02}	5.5242×10^{-02}	$\mathbf{3.5564 \times 10^{-02}}$
80	3.6693×10^{-02}	5.6178×10^{-02}	$\mathbf{3.6693 \times 10^{-02}}$
85	3.7868×10^{-02}	5.4755×10^{-02}	$\mathbf{3.7868 \times 10^{-02}}$
90	3.9025×10^{-02}	6.5662×10^{-02}	$\mathbf{3.9025 \times 10^{-02}}$
95	4.0027×10^{-02}	6.2286×10^{-02}	$\mathbf{4.0027 \times 10^{-02}}$
100	4.1095×10^{-02}	6.8882×10^{-02}	$\mathbf{4.1095 \times 10^{-02}}$

TABLE VI
ACCURACY NOISE DENSITY $\varepsilon = 1.0 \times 10^{+01}$

Dimension n	ϵ_{SVD}	ϵ_{LMI}	$\epsilon_{\text{Proposed}}$
5	$2.9183 \times 10^{+00}$	$4.7148 \times 10^{+00}$	$\mathbf{2.9183 \times 10^{+00}}$
10	$4.1216 \times 10^{+00}$	$6.9657 \times 10^{+00}$	$\mathbf{4.1216 \times 10^{+00}}$
15	$5.0425 \times 10^{+00}$	$8.8207 \times 10^{+00}$	$\mathbf{5.0425 \times 10^{+00}}$
20	$5.8215 \times 10^{+00}$	$9.2720 \times 10^{+00}$	$\mathbf{5.8215 \times 10^{+00}}$
25	$6.5206 \times 10^{+00}$	$1.1008 \times 10^{+01}$	$\mathbf{6.5206 \times 10^{+00}}$
30	$7.1352 \times 10^{+00}$	$1.2353 \times 10^{+01}$	$\mathbf{7.1352 \times 10^{+00}}$
35	$7.7093 \times 10^{+00}$	$1.3150 \times 10^{+01}$	$\mathbf{7.7093 \times 10^{+00}}$
40	$8.2448 \times 10^{+00}$	$1.3918 \times 10^{+01}$	$\mathbf{8.2448 \times 10^{+00}}$
45	$8.7331 \times 10^{+00}$	$1.2899 \times 10^{+01}$	$\mathbf{8.7331 \times 10^{+00}}$
50	$9.2070 \times 10^{+00}$	$1.3756 \times 10^{+01}$	$\mathbf{9.2070 \times 10^{+00}}$
55	$9.6519 \times 10^{+00}$	$1.5520 \times 10^{+01}$	$\mathbf{9.6519 \times 10^{+00}}$
60	$1.0085 \times 10^{+01}$	$1.5606 \times 10^{+01}$	$\mathbf{1.0085 \times 10^{+01}}$
65	$1.0494 \times 10^{+01}$	$1.7121 \times 10^{+01}$	$\mathbf{1.0494 \times 10^{+01}}$
70	$1.0898 \times 10^{+01}$	$1.5725 \times 10^{+01}$	$\mathbf{1.0898 \times 10^{+01}}$
75	$1.1281 \times 10^{+01}$	$1.8923 \times 10^{+01}$	$\mathbf{1.1281 \times 10^{+01}}$
80	$1.1639 \times 10^{+01}$	$1.8249 \times 10^{+01}$	$\mathbf{1.1639 \times 10^{+01}}$
85	$1.2020 \times 10^{+01}$	$1.7407 \times 10^{+01}$	$\mathbf{1.2020 \times 10^{+01}}$
90	$1.2344 \times 10^{+01}$	$1.8864 \times 10^{+01}$	$\mathbf{1.2344 \times 10^{+01}}$
95	$1.2693 \times 10^{+01}$	$1.9491 \times 10^{+01}$	$\mathbf{1.2693 \times 10^{+01}}$
100	$1.3016 \times 10^{+01}$	$1.9967 \times 10^{+01}$	$\mathbf{1.3016 \times 10^{+01}}$

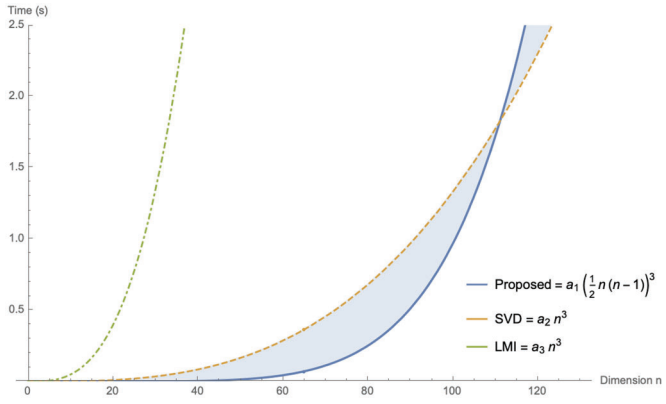


Fig. 1. Time complexity performances of various algorithms.

However, when n becomes larger, the matrix manipulations of the proposed method also turn to be more and more time-consuming. This leads to the deceasing ratio of $t_{\text{SVD}}/t_{\text{Proposed}}$. That is to say, for very large n, SVD will be faster and the LMI will be even more complex than the proposed method, as indicated by $t_{\text{LMI}}/t_{\text{Proposed}}$ (also see Fig. 2). Note that there are very few cases that n is larger than 100, the proposed method is more practical than SVD in most cases since it also owns information of covariances.

C. Application: Covariance-Preserving Interpolation on Special Euclidean Group $SE(n)$

The special Euclidean group $SE(n)$ is a simultaneous precise description of the rotation and translation, which has been extensively used in rigid-body state estimation and control [23], [50], [51]. Also, in the field of robotics, $SE(n)$ plays important roles in camera egomotion estimation, hand-eye calibration, motion planning, etc. [26], [31], [52]. However, in engineering applications, the measurements on $SE(n)$ are usually restricted by low sampling frequency and asynchronous data transmission. Therefore, the interpolated result between two successive measurements on $SE(n)$ will be vital to such systems. The interpolation of $SE(n)$ is sometimes dependent on the interpolation on $SO(n)$. Belta and Kumar studied the interpolation on $SE(3)$ using SVD, where the rotation and translation are simultaneously interpolated for optimal approximation with error of $SE(3)$ geodesics. Because of the internal nonlinearity of SVD, it is very hard for the users to obtain the closed-form information of covariance of the interpolated results. The probabilistic descriptions of interpolation would be of great significance for a later covariance-required application e.g., a Kalman filter on the Lie group [53], [54]. Therefore, our next task is to present a new framework for such a purpose.

In [55], Thomas gave an approximation from $SE(3)$ to $SO(4)$, which is later improved in [31]. We have proposed a new generalization such that the following mapping $\mathcal{F}$ from $SE(n)$ to $SO(n+1)$, namely, $SE(n)++$ [56]:

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \in SE(n) \quad \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\longleftrightarrow}}$$
$$\boldsymbol{R}_{\boldsymbol{T},SO(n+1)} = \begin{pmatrix} \boldsymbol{R} & \varepsilon \boldsymbol{t} \\ -\varepsilon \boldsymbol{t}^{\top} \boldsymbol{R} & 1 \end{pmatrix} \tag{48}$$

holds. Note that here $\boldsymbol{R}_{\boldsymbol{T},SO(n+1)}$ is not strictly on $SO(n)$ and should be orthonormalized to $\boldsymbol{X}_{\boldsymbol{R}}$ before further computation

$$\underset{\boldsymbol{X}_{\boldsymbol{R}} \in SO(n+1)}{\arg\max} \ \operatorname{tr}\left[\boldsymbol{X}_{\boldsymbol{R}} \boldsymbol{R}_{\boldsymbol{T},SO(n+1)}^{\top}\right] \tag{49}$$

which can be solved via the proposed algorithm by forming the following vector pairs [57], [58]:

$$\begin{cases} \boldsymbol{b}_1 = \boldsymbol{R}_{\boldsymbol{T},SO(n+1),1} \\ \boldsymbol{b}_2 = \boldsymbol{R}_{\boldsymbol{T},SO(n+1),2} \\ \vdots \\ \boldsymbol{b}_n = \boldsymbol{R}_{\boldsymbol{T},SO(n+1),n} \end{cases}, \begin{cases} \boldsymbol{r}_1 = (1, 0, \ldots, 0)^{\top} \\ \boldsymbol{r}_2 = (0, 1, \ldots, 0)^{\top} \\ \vdots \\ \boldsymbol{r}_n = (0, 0, \ldots, 1)^{\top} \end{cases} \tag{50}$$

with column vectors of $\boldsymbol{R}_{\boldsymbol{T},SO(n+1)}$ being

$$\boldsymbol{R}_{\boldsymbol{T},SO(n+1)} = \left[\boldsymbol{R}_{\boldsymbol{T},SO(n+1),1}, \boldsymbol{R}_{\boldsymbol{T},SO(n+1),2}, \ldots, \boldsymbol{R}_{\boldsymbol{T},SO(n+1),n}\right]. \tag{51}$$

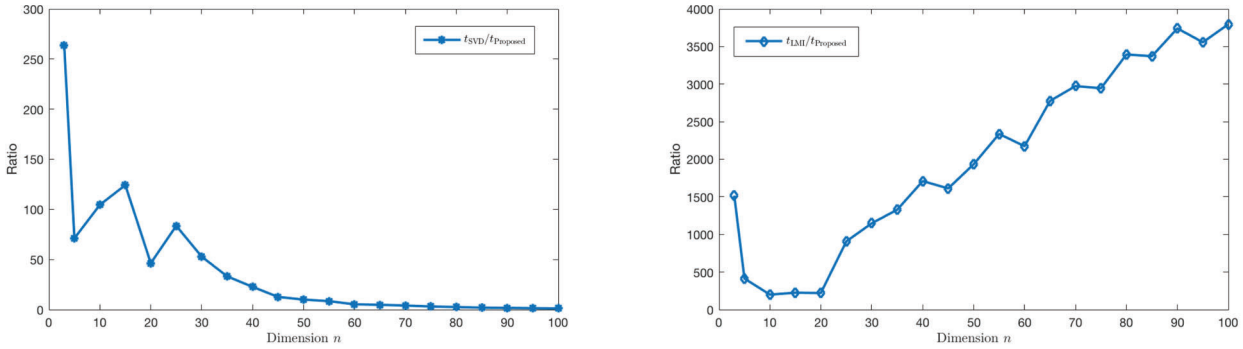


Fig. 2. Ratios of computation time from SVD, LMI, and the proposed method.

TABLE VII
COMPUTATION TIME

Dimension n	t_{SVD}	t_{LMI}	t_{Proposed}	$t_{\mathrm{SVD}}/t_{\mathrm{Proposed}}$	$t_{\mathrm{LMI}}/t_{\mathrm{Proposed}}$
5	$5.09175700 \times 10^{-03}$ sec	$2.95878440 \times 10^{-02}$ sec	$\mathbf{7.14850000 \times 10^{-05}}$ sec	$7.12283276 \times 10^{+01}$	$4.13902833 \times 10^{+02}$
10	$1.86192880 \times 10^{-02}$ sec	$3.57405530 \times 10^{-02}$ sec	$\mathbf{1.78072000 \times 10^{-04}}$ sec	$1.04560447 \times 10^{+02}$	$2.00708438 \times 10^{+02}$
15	$2.91524970 \times 10^{-02}$ sec	$5.29616330 \times 10^{-02}$ sec	$\mathbf{2.34585000 \times 10^{-04}}$ sec	$1.24272639 \times 10^{+02}$	$2.25767347 \times 10^{+02}$
20	$5.16656850 \times 10^{-02}$ sec	$2.46701489 \times 10^{-01}$ sec	$\mathbf{1.11337100 \times 10^{-03}}$ sec	$4.64047339 \times 10^{+01}$	$2.21580667 \times 10^{+02}$
25	$7.20797970 \times 10^{-02}$ sec	$7.83117401 \times 10^{-01}$ sec	$\mathbf{8.61778000 \times 10^{-04}}$ sec	$8.36407950 \times 10^{+01}$	$9.08722897 \times 10^{+02}$
30	$1.10495224 \times 10^{-01}$ sec	$2.40003185 \times 10^{+00}$ sec	$\mathbf{2.08869100 \times 10^{-03}}$ sec	$5.29016614 \times 10^{+01}$	$1.14906027 \times 10^{+03}$
35	$1.51141937 \times 10^{-01}$ sec	$6.04819777 \times 10^{+00}$ sec	$\mathbf{4.54621700 \times 10^{-03}}$ sec	$3.32456495 \times 10^{+01}$	$1.33038035 \times 10^{+03}$
40	$2.03090954 \times 10^{-01}$ sec	$1.52671507 \times 10^{+01}$ sec	$\mathbf{8.92617900 \times 10^{-03}}$ sec	$2.27522834 \times 10^{+01}$	$1.71037918 \times 10^{+03}$
45	$2.56202188 \times 10^{-01}$ sec	$3.24307563 \times 10^{+01}$ sec	$\mathbf{2.01093880 \times 10^{-02}}$ sec	$1.27404269 \times 10^{+01}$	$1.61271722 \times 10^{+03}$
50	$3.12731122 \times 10^{-01}$ sec	$6.00155415 \times 10^{+01}$ sec	$\mathbf{3.10260630 \times 10^{-02}}$ sec	$1.00796263 \times 10^{+01}$	$1.93435891 \times 10^{+03}$
55	$3.81157892 \times 10^{-01}$ sec	$1.04683072 \times 10^{+02}$ sec	$\mathbf{4.47886660 \times 10^{-02}}$ sec	$8.51014165 \times 10^{+00}$	$2.33726701 \times 10^{+03}$
60	$4.78917751 \times 10^{-01}$ sec	$1.94233782 \times 10^{+02}$ sec	$\mathbf{8.93274220 \times 10^{-02}}$ sec	$5.36137437 \times 10^{+00}$	$2.17440264 \times 10^{+03}$
65	$5.47775943 \times 10^{-01}$ sec	$3.12991523 \times 10^{+02}$ sec	$\mathbf{1.12696783 \times 10^{-01}}$ sec	$4.86061739 \times 10^{+00}$	$2.77728888 \times 10^{+03}$
70	$6.46396852 \times 10^{-01}$ sec	$4.61347369 \times 10^{+02}$ sec	$\mathbf{1.55126971 \times 10^{-01}}$ sec	$4.16688889 \times 10^{+00}$	$2.97399843 \times 10^{+03}$
75	$7.63718827 \times 10^{-01}$ sec	$7.12493501 \times 10^{+02}$ sec	$\mathbf{2.41905543 \times 10^{-01}}$ sec	$3.15709519 \times 10^{+00}$	$2.94533764 \times 10^{+03}$
80	$8.53510834 \times 10^{-01}$ sec	$1.10820564 \times 10^{+03}$ sec	$\mathbf{3.26491385 \times 10^{-01}}$ sec	$2.61419098 \times 10^{+00}$	$3.39428753 \times 10^{+03}$
85	$1.00388324 \times 10^{+00}$ sec	$1.68530646 \times 10^{+03}$ sec	$\mathbf{4.99838141 \times 10^{-01}}$ sec	$2.00841664 \times 10^{+00}$	$3.37170439 \times 10^{+03}$
90	$1.19993136 \times 10^{+00}$ sec	$2.46278443 \times 10^{+03}$ sec	$\mathbf{6.58335981 \times 10^{-01}}$ sec	$1.82267322 \times 10^{+00}$	$3.74092333 \times 10^{+03}$
95	$1.27744737 \times 10^{+00}$ sec	$3.06590801 \times 10^{+03}$ sec	$\mathbf{8.61907923 \times 10^{-01}}$ sec	$1.48211582 \times 10^{+00}$	$3.55711779 \times 10^{+03}$
100	$1.40629819 \times 10^{+00}$ sec	$4.35698895 \times 10^{+03}$ sec	$\mathbf{1.14776548 \times 10^{+00}}$ sec	$1.22524872 \times 10^{+00}$	$3.79606203 \times 10^{+03}$

The covariance of $\boldsymbol{X}_{\boldsymbol{R}}$ is then given by the (40). For a series of mapped $SO(n+1)$ measurements mapped from $SE(n)$ $\boldsymbol{X}_{\boldsymbol{R},1}, \boldsymbol{X}_{\boldsymbol{R},1}, \ldots, \boldsymbol{X}_{\boldsymbol{R},k}$, with their timestamps $\tau_1, \tau_2, \ldots, \tau_k$, we can interpolate between the time instants $k-1$ and k by

$$\underset{\boldsymbol{X}_{\boldsymbol{R},k-1|k} \in SO(n+1)}{\arg\max} \operatorname{tr}\Big\{\boldsymbol{X}_{\boldsymbol{R},k-1|k}\big[\varrho \boldsymbol{X}_{\boldsymbol{R},k-1} + (1-\varrho)\boldsymbol{X}_{\boldsymbol{R},k}\big]^{\top}\Big\} \tag{52}$$

where ϱ is the relative weight such that

$$\varrho = \frac{\tau_k - \tau_{k-1|k}}{\tau_k - \tau_{k-1}} \in (0,1). \tag{53}$$

Likewise, by setting

$$\begin{aligned} (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n) &= \big[\varrho \boldsymbol{X}_{\boldsymbol{R},k-1} + (1-\varrho)\boldsymbol{X}_{\boldsymbol{R},k}\big] \\ &\begin{cases} \boldsymbol{r}_1 = (1,0,\ldots,0)^{\top} \\ \boldsymbol{r}_2 = (0,1,\ldots,0)^{\top} \\ \vdots \\ \boldsymbol{r}_n = (0,0,\ldots,1)^{\top} \end{cases} \end{aligned} \tag{54}$$

one is able to compute the interpolated result on $SO(n+1)$. The covariance of $\boldsymbol{X}_{\boldsymbol{R},k-1|k}$ can also be calculated with (40). Then, $\boldsymbol{X}_{\boldsymbol{R},k-1|k}$ should be inversed from $SO(n+1)$ back to $SE(n)$ by the mapping $\mathcal{F}^{-1}$. The rest steps for generating covariances of the rotational and translational parts can be referred to [31].

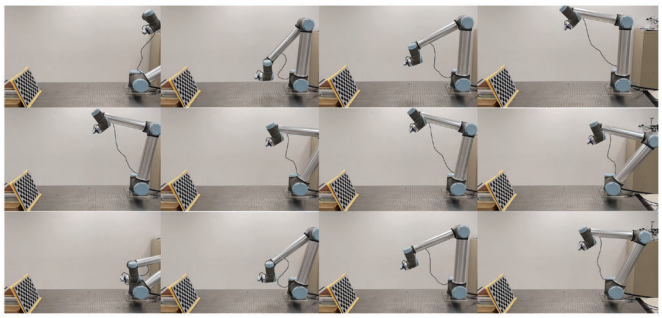


Fig. 3. Generated motion of the UR10 robotic arm.

We use the UR10 robotic arm to perform a generated series of motion for over 150 s while an Intel Realsense D435i camera is attached to the end-effector of this arm (see Fig. 3). A standard 12×9 chessboard is performed in the experimental environment for distinct feature extraction. There are inertial and visual outputs from the D435i but it has a low frequency for the fisheye camera (30 fps). Therefore, a visual-inertial navigation system is employed to estimate the motion of the

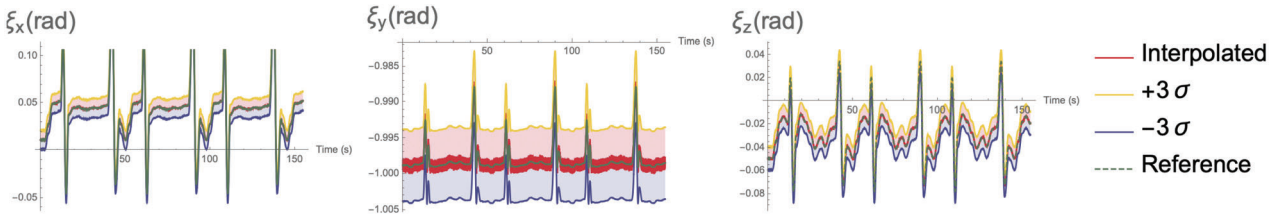


Fig. 4. Interpolated results and their 3σ bounds for the eigenaxis $\boldsymbol{\xi}$.

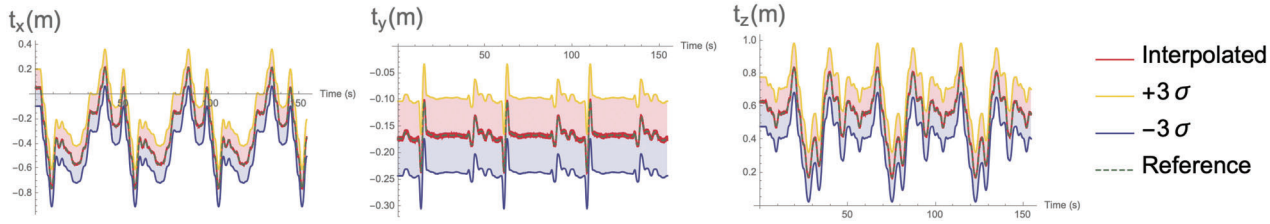


Fig. 5. Interpolated results and their 3σ bounds for the translation $\boldsymbol{t}$.

end-effector. Note that the hand-eye calibration between the installation frame of the end-effector and the sensor frame of the Realsense camera has been performed before experimentation so that the reference transformation from the readings of the robotic arm can be mapped precisely into the camera frame [31].

The 1σ uncertainty of the camera measurement is set to be within 1 pixel while the in-run covariances of the inertial measurement unit (IMU) are

$$\begin{aligned} \boldsymbol{\Sigma}_{\text{gyro}} &= 1.34 \times 10^{-5} \boldsymbol{I} \text{ rad}^2 \\ \boldsymbol{\Sigma}_{\text{accel}} &= 2.02 \times 10^{-3} \boldsymbol{I} \text{ (m/s}^2)^2 \end{aligned} \tag{55}$$

where gyro and accel denote the gyroscope and accelerometer, respectively, and the covariances are produced according to a live Allan variance test for 1 h [59]. For each visual measurement, the visual-inertial state estimator computes the state estimation along with its covariance by the IMU preintegration [60]. These results have low sampling frequencies so they are interpolated uniformly to 100 Hz so the reference information provided by the robotic arm could be compared with. The proposed algorithm is invoked to conduct such interpolation on $SE(3)$, which follows the equations listed in this section. The interpolated results along with their 3σ bounds are depicted in Figs. 4 and 5. In Fig. 4, $\boldsymbol{\xi} = (\xi_x, \xi_y, \xi_z)^\top$ denotes the eigenaxis of a 3×3 rotation matrix $\boldsymbol{R}$ so that $\exp(\boldsymbol{\xi}_\times) = \boldsymbol{R}$ and $\boldsymbol{R}\boldsymbol{\xi} = \boldsymbol{\xi}$. The covariance $\boldsymbol{\Sigma}_{\boldsymbol{R}}$ actually reflects the uncertainty of $\boldsymbol{\xi}$ [61]. The translation in this case is in the 3-D form such that $\boldsymbol{t} = (t_x, t_y, t_z)^\top$. We are able to observe that the robotic manipulator mainly moves along its x and z axes. It is observed that using the proposed method for the interpolation on $SE(3)$, we are able to recover the original motion into a more dense one without loss of probabilistic propagation. This is reflected in Figs. 4 and 5 that all these interpolated results are well within the 3σ bounds. As we uses the mapping from $SE(3)$ to $SO(4)$ for such interpolation, the proposed method is validated to be reliable and robust on $SO(4)$ and can achieve further state estimation tasks on $SO(n)$ requiring uncertainty descriptions.

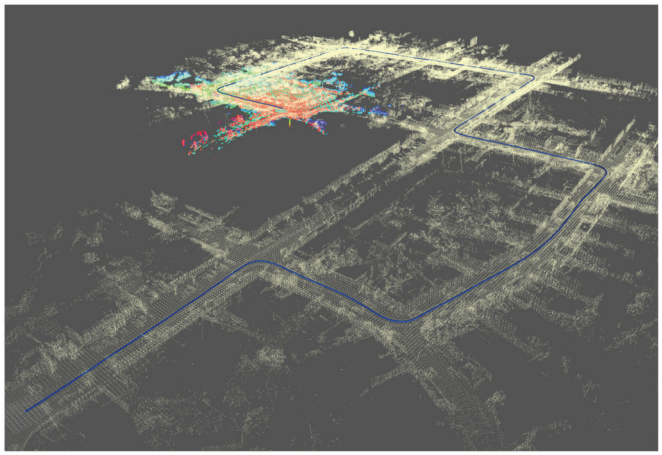


Fig. 6. Mapping result of the proposed Lidar registration algorithm using first 125 s of the KITTI dataset `kitti_2011_10_03_drive_0027_synced` [62]. The blue line denotes the vehicle trajectory. The blue and green points denote the local map while red points denotes the matching result. The white points represent the reconstructed dense map after covariance-aided pose graph optimization.

D. Application: Covariance-Aided Lidar Mapping

We use the KITTI dataset `kitti_2011_10_03_drive_0027_synced` [62] to demonstrate covariance-aided Lidar mapping using the results from the proposed method. The KITTI dataset contains some data from a 64-beam Velodyne HDL-64E Lidar. The Lidar has the measurement specification of 0.09° angular resolution, 2-cm distance accuracy, which can be used for setting the measurement covariance for further propagation. We evaluate the trajectory errors by comparing the results with the ground-truth data in KITTI dataset. Since the entire trajectory is lengthy, we only visualize the first 125-s results in Fig. 6. In these results, the frame-to-frame matching is conducted via formula in (15). The covariance results of rotation and translation are then generated and transferred to a further pose graph optimization (PGO). Here, we select the SE-Sync method [63] for globally optimal PGO computation. Therefore, the numerical computation of PGO is guaranteed

TABLE VIII
RMSEs OF THE ESTIMATED TRAJECTORIES

Direction	Barczyk et al. [37]	Censi [64]	Proposed
X	2.472284 m	2.536159 m	**2.421336** m
Y	2.034533 m	**2.007825** m	2.016879 m
Z	1.9824760 m	1.992351 m	**1.972403** m

to be optimal. Since successive frames are close to each other, the correspondence can be easily found by the random sample consensus (RANSAC), starting from an identity pose. Thus, we do not consider the uncertainty of the point correspondences in this work, i.e., it will be regarded as deterministic and only pose covariance in (40) and (41) will be taken into account. We select two representatives dealing with covariance of ICP, i.e., works in [37] and [64], to make comparisons. The trajectory root mean squared errors (RMSEs) are shown in Table VIII.

The best candidate in each direction is marked bold in Table VIII. It can be observed that although Censi's method outperforms the proposed method on *y*-axis, the overall trajectory accuracy considering *XYZ*-axes of the proposed method is the best. The reason is that both [37] and [64] estimate the covariance of ICP by taking the Hessian and approximates the covariance of the least-square in first-order, which will inevitably bring about accuracy loss. The proposed method does not need any numerical optimization and is explicit so the closed-form covariance can be derived in an analytical manner as well. These analytical forms give the rise to the covariance accuracy, which has been indirectly revealed in the accuracy of the localization after PGO.

IV. CONCLUSION

The n-dimensional rigid registration problem is revisited in this article. It is shown that using the Caylay transformation, we are able to establish a linear framework for computing the fundamental parameters. Related covariance analysis of these parameters along with recovered rotation and translation can be conducted flexibly due to the existence of the proposed linear solution. It is verified that the proposed method is slower than SVD in very high dimensions ($n > 100$) for a modern computer but SVD can not provide probabilistic information of the estimates. Finally, we propose a new method for interpolating measurements on the special Euclidean group $SE(3)$, showing that the proposed algorithm can well handle the high-dimensional rotation orthonormalization and interpolation with uncertainty descriptions. The current drawback of the proposed method is evident that it consumes too many computational resources for cases with very high dimensions. Future efforts should be paid to seek a more computationally efficient numerical framework for fast inverse or pseudo inverse of arbitrary matrices. Source codes of this article will be made open-access on `https://github.com/zarathustr/GLnR`. The video of this work is presented on `https://youtu.be/BwfjQ9ZAyl4`.

ACKNOWLEDGMENT

The authors would like to thank Prof. Xiao-Jun Wu from Jiangnan University for providing some platforms for experimentation.

REFERENCES

[1] M. Liu, "Robotic online path planning on point cloud," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1217–1228, May 2016.

[2] H. Xie, W.-L. Li, Z.-P. Yin, and H. Ding, "Variance-minimization iterative matching method for free-form surfaces—Part I: Theory and method," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1181–1191, Jul. 2019.

[3] H. Xie, W.-L. Li, Z.-P. Yin, and H. Ding, "Variance-minimization iterative matching method for free-form surfaces—Part II: Experiment and analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1192–1204, Jul. 2019.

[4] P. Paloj, K. Maatta, and J. Kostamovaara, "Integrated time-of-flight laser radar," *IEEE Trans. Instrum. Meas.*, vol. 46, no. 4, pp. 996–999, Aug. 1997.

[5] M. Liu and R. Siegwart, "Topological mapping and scene recognition with lightweight color descriptors for an omnidirectional camera," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 310–324, Apr. 2014.

[6] L. Bai, X. Yang, and H. Gao, "Nonrigid point set registration by preserving local connectivity," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 826–835, Mar. 2018.

[7] L. Li, M. Yang, C. Wang, and B. Wang, "Robust point set registration using signature quadratic form distance," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 2097–2109, May 2020, doi: 10.1109/TCYB.2018.2845745.

[8] O. Tahri, H. Araujo, Y. Mezouar, and F. Chaumette, "Efficient iterative pose estimation using an invariant to rotations," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 199–207, Feb. 2014.

[9] P. Miraldo, H. Araujo, and N. Goncalves, "Pose estimation for general cameras using lines," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2156–2164, Oct. 2015.

[10] J. Wu, Z. Zhou, B. Gao, R. Li, Y. Cheng, and H. Fourati, "Fast linear quaternion attitude estimator using vector observations," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 307–319, Jan. 2018.

[11] F. Aghili and C.-Y. Su, "Robust relative navigation by integration of ICP and adaptive Kalman filter using laser scanner and IMU," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 4, pp. 2015–2026, Aug. 2016.

[12] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[13] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[14] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.

[15] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.

[16] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. America A*, vol. 4, no. 4, p. 629, 1987.

[17] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP Image Understand.*, vol. 54, no. 3, pp. 358–367, 1991.

[18] J. Wu, M. Liu, Z. Zhou, and R. Li, "Fast symbolic 3-D registration solution," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 761–770, Apr. 2020.

[19] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3-D ICP point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, Nov. 2016.

[20] R. Sanyal, S. M. Ahmed, M. Jaiswal, and K. N. Chaudhury, "A scalable ADMM algorithm for rigid registration," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1453–1457, Oct. 2017.

[21] R. Sanyal, M. Jaiswal, and K. N. Chaudhury, "On a registration-based approach to sensor network localization," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5357–5367, Oct. 2017.

[22] H. A. Hashim, L. J. Brown, and K. McIsaac, "Nonlinear pose filters on the special Euclidean group SE(3) with guaranteed transient and steady-state performance," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 5, pp. 2949–2962, May 2021, doi: 10.1109/TSMC.2019.2920114.

[23] H. A. Hashim, L. J. Brown, and K. McIsaac, "Nonlinear stochastic attitude filters on the special orthogonal group 3: Ito and stratonovich," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 9, pp. 1853–1865, Sep. 2019, doi: 10.1109/TSMC.2018.2870290.

[24] J. Markdahl and X. Hu, "Exact solutions to a class of feedback systems on SO(n)," *Automatica*, vol. 63, pp. 138–147, Jan. 2016.

[25] J. Saunderson, P. A. Passilo, and A. S. Willsky, "Semidefinite descriptions of the convex hull of rotation matrices," *SIAM J. Optim.*, vol. 25, no. 3, pp. 1314–1343, 2015.

[26] J. D. Biggs and H. C. Henninger, "Motion planning on a class of 6-D Lie groups via a covering map," *IEEE Trans. Autom. Control*, vol. 64, no. 9, pp. 3544–3554, Sep. 2019, doi: 10.1109/TAC.2018.2885241.

[27] S. Du, N. Zheng, S. Ying, and J. Liu, "Affine iterative closest point algorithm for point set registration," *Pattern Recognit. Lett.*, vol. 31, no. 9, pp. 791–799, 2010.

[28] A. H. J. de Ruiter and J. R. Forbes, "On the solution of Wahba's problem on SO(n)," *J. Astronaut. Sci.*, vol. 60, pp. 1–31, Dec. 2014.

[29] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Trans. Autom. Control*, vol. 53, no. 5, pp. 1203–1218, Jun. 2008.

[30] S. Berkane and A. Tayebi, "On the design of attitude complementary filters on SO(3)," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 880–887, Mar. 2018.

[31] J. Wu, Y. Sun, M. Wang, and M. Liu, "Hand-eye calibration: 4-D procrustes analysis approach," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 2966–2981, Jun. 2020.

[32] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3-D robotics hand/eye calibration," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 345–358, Jun. 1989.

[33] H. Zhuang and Y. C. Shiu, "A noise-tolerant algorithm for robotic hand-eye calibration with or without sensor orientation measurement," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 4, pp. 1168–1175, Jul./Aug. 1993.

[34] H. Zhuang, Z. S. Roth, and R. Sudhakar, "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form AX = YB," *IEEE Trans. Robot. Autom.*, vol. 10, no. 4, pp. 549–554, Aug. 1994.

[35] N. Andreff, R. Horaud, and B. Espiau, "Robot hand-eye calibration using structure-from-motion," *Int. J. Robot. Res.*, vol. 20, no. 3, pp. 228–248, 2001.

[36] D. Condurache and A. Burlacu, "Orthogonal dual tensor method for solving the AX = XB sensor calibration problem," *Mech. Mach. Theory*, vol. 104, pp. 382–404, Oct. 2016.

[37] M. Barczyk, S. Bonnabel, J.-E. Deschaud, and F. Goulette, "Invariant EKF design for scan matching-aided localization," *IEEE Trans. Control Syst. Tech.*, vol. 23, no. 6, pp. 2440–2448, Nov. 2015.

[38] J. Wu, Z. Zhou, H. Fourati, R. Li, and M. Liu, "Generalized linear quaternion complementary filter for attitude estimation from multisensor observations: An optimization approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1330–1343, Jul. 2019.

[39] V. Mohammadi and M. Dehghan, "A divergence-free generalized moving least squares approximation with its application," *Appl. Numer. Math.*, vol. 162, pp. 374–404, Apr. 2021.

[40] W. S. Massey, "Cross products of vectors in higher dimensional Euclidean spaces," *Amer. Math. Monthly*, vol. 90, no. 10, pp. 697–701, 1983.

[41] D. Mortari, F. L. Markley, and P. Singla, "Optimal linear attitude estimator," *AIAA J. Guid. Control Dyn.*, vol. 30, no. 6, pp. 1619–1627, 2007.

[42] P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic Complexity Theory*, vol. 315. London, U.K.: Springer, 2013.

[43] J. Wu, Z. Zhou, H. Fourati, and M. Liu, "Recursive linear continuous quaternion attitude estimator from vector observations," *IET Radar Sonar Nav.*, vol. 12, no. 11, pp. 1196–1207, 2018.

[44] F. L. Markley and Y. Cheng, "Wahba's problem with one dominant observation," *AIAA J. Guid. Control Dyn.*, vol. 41, no. 9, pp. 1–6, 2018.

[45] D. Gebre-Egziabher and G. H. Elkaim, "MAV attitude determination by vector matching," *IEEE Trans. Aerosp. Elect. Syst.*, vol. 44, no. 3, pp. 1012–1028, Jul. 2008.

[46] S. Du, N. Zheng, G. Meng, and Z. Yuan, "Affine registration of point sets using ICP and ICA," *IEEE Signal Process. Lett.*, vol. 15, no. 2, pp. 689–692, Nov. 2008.

[47] J. Heller and T. Pajdla, "GpoSolver: A MATLAB/C++ toolbox for global polynomial optimization," *Optim. Methods Softw.*, vol. 31, no. 2, pp. 405–434, 2016.

[48] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *J. Astronaut. Sci.*, vol. 36, no. 3, pp. 245–258, 1988.

[49] J. R. Forbes and A. H. J. de Ruiter, "Linear-matrix-inequality-based solution to Wahba's problem," *AIAA J. Guid. Control Dyn.*, vol. 38, no. 1, pp. 147–151, 2015.

[50] M. Wang and A. Tayebi, "Hybrid pose and velocity-bias estimation on SE(3) using inertial and landmark measurements," *IEEE Trans. Autom. Control*, vol. 64, no. 8, pp. 3399–3406, Aug. 2019, doi: 10.1109/TAC.2018.2879766.

[51] G. Baldwin, R. Mahony, J. Trumpf, T. Hamel, and T. Cheviron, "Complementary filter design on the special Euclidean group SE(3)," in *Proc. IEEE ECC*, 2007, pp. 3763–3770.

[52] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 930–943, Aug. 2003.

[53] S. Bonnabel, P. Martin, and P. Rouchon, "Symmetry-preserving observers," *IEEE Trans. Autom. Control*, vol. 53, no. 11, pp. 2514–2526, Dec. 2008.

[54] A. H. J. de Ruiter and J. R. Forbes, "Discrete-time SO(n)-constrained Kalman filtering," *AIAA J. Guid. Control Dyn.*, vol. 40, no. 1, pp. 28–37, 2017.

[55] F. Thomas, "Approaching dual quaternions from matrix algebra," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1037–1048, Oct. 2014.

[56] J. Wu, M. Liu, Y. Huang, C. Jin, Y. Wu, and C. Yu, "SE(n)++: An efficient solution to multiple pose estimation problems," *IEEE Trans. Cybern.*, early access, Sep. 2, 2020, doi: 10.1109/TCYB.2020.3015039.

[57] I. Y. Bar-Itzhack, "New method for extracting the quaternion from a rotation matrix," *AIAA J. Guid. Control Dyn.*, vol. 23, no. 6, pp. 1085–1087, 2000.

[58] J. Wu, "Optimal continuous unit quaternions from rotation matrices," *AIAA J. Guid. Control Dyn.*, vol. 42, no. 4, pp. 919–922, 2019.

[59] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of MEMS based inertial sensors using Allan variance," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 140–149, Jan. 2008.

[60] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[61] H. Nguyen and Q.-C. Pham, "On the covariance of X in AX = XB," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1651–1658, Dec. 2018.

[62] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[63] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 95–125, 2019.

[64] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proc. IEEE ICRA*, 2007, pp. 3167–3172.

Jin Wu (Member, IEEE) was born in Zhenjiang, Jiangsu, China, in May 1994. He received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, supervised by Prof. M. Liu.

From 2012 to 2018, he was in the UAV industry. In 2013, he was a visiting student with Groep T, KU Leuven, Leuven, Belgium. He was with Tencent Robotics X, Shenzhen, China, from 2019 to 2020.

Miaomiao Wang (Member, IEEE) received the B.Eng. degree in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2013, the M.Sc. degree in control engineering from Lakehead University, Thunder Bay, ON, Canada, in 2015, and the Ph.D. degree in robotics and control engineering from Western University, London, ON, Canada, in 2020.

He is currently a Postdoctoral Associate with the Department of Electrical and Computer Engineering, Western University. His current research interests include the areas of nonlinear state estimation and geometric control with applications to autonomous robotics.

Hassen Fourati received the B.Eng. degree in electrical engineering from the National Engineering School of Sfax, Sfax, Tunisia, in 2006, the master's degree in automated systems and control from University Claude Bernard, Lyon, France, in 2007, and the Ph.D. degree in automatic control from the University of Strasbourg, Strasbourg, France, in 2010.

He is currently an Associate Professor of Electrical Engineering and Computer Science with University Grenoble Alpes, Grenoble, France. His research interests include nonlinear filtering and estimation and multisensor fusion with applications in navigation, robotics, and traffic management.

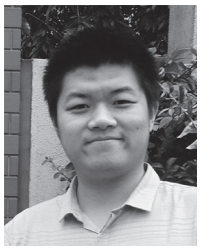
Hui Li (Member, IEEE) received the B.Sc. and Ph.D. degrees from the School of Internet of Things Engineering, Jiangnan University, Wuxi, China, in 2015 and 2022, respectively.

He is currently a Lecturer with the School of Artificial Intelligence and Computer Science, Jiangnan University. His research interests include image fusion and multimodal visual information processing.

Yilong Zhu was born in 1994. He received the B.Sc. degree from Harbin Institute of Technology, Harbin, China, in 2017, and the M.S. degree in electrical engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2018, supervised by Prof. M. Liu, where he is currently pursuing the Ph.D. degree with RAM-LAB and his research mainly focuses on LiDAR and sensor fusion for robotics.

Chengxi Zhang (Member, IEEE) was born in Shandong, China, in February 1990. He received the B.S. and M.S. degrees in microelectronics and solid-state electronics from Harbin Institute of Technology, Harbin, China, in 2012 and 2015, respectively, and the Ph.D. degree in control science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019.

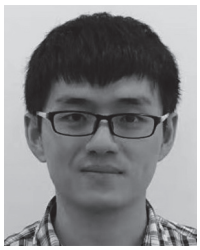
He was a Postdoctoral Fellow with the School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen Campus), Shenzhen, China, from 2020 to 2022. His research interests are embedded system software and hardware design, information fusion, and control theory.



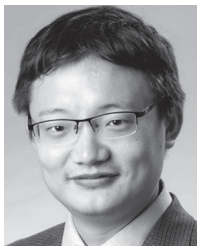
Yi Jiang (Member, IEEE) was born in Ezhou, Hubei, China. He received the B.E., M.S., and Ph.D. degrees from Northeastern University, Shenyang, Liaoning, China.

He is currently a Postdoctoral Fellow with the City University of Hong Kong, Hong Kong. His research interests include networked control systems, industrial process operational control, reinforcement learning, and event-triggered control.

Dr. Jiang is the recipient of the Excellent Doctoral Dissertation Award from Chinese Association of Automation in 2021. He is an Associate Editor of *Advanced Control for Applications: Engineering and Industrial Systems* (Wiley).



Xiangcheng Hu was born in Macheng, Hubei, China, in 1995. He received the B.Sc. degree from the North University of China, Taiyuan, China, in 2017, and the M.S. degree from Beihang University, Beijing, China, in 2020. He is currently pursuing the Ph.D. degree with the Robotics and Multiperception Laboratory, The Hong Kong University of Science and Technology, Hong Kong, supervised by Prof. M. Liu, where his research mainly focuses on LiDAR mapping and sensor fusion for robotics.



Ming Liu (Senior Member, IEEE) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005, and the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zürich, Zürich, Switzerland, in 2013, supervised by Prof. R. Siegwart.

He is currently with the Electronic and Computer Engineering, Computer Science and Engineering Department, Robotics Institute, The Hong Kong University of Science and Technology, Hong Kong, as an Associate Professor. He is currently the Chairman of Shenzhen Unity Drive Inc., Shenzhen, China. He has published many popular papers in top robotics journals, including IEEE TRANSACTIONS ON ROBOTICS, *International Journal of Robotics Research*, and *IEEE Robotics and Automation Magazine*. His research interests include dynamic environment modeling, deep learning for robotics, 3-D mapping, machine learning, and visual control.

Dr. Liu was a recipient of the EMAV 2009 (second place) and two awards from IARC 2014, as a Team Member, the Best Student Paper Award as the first author for MFI 2012, the Best Paper Award in Information for IEEE ICIA 2013 as the first author, the Best Paper Award Finalists as the coauthor, the Best RoboCup Paper Award for IROS 2013, and the Best Conference Paper Award for IEEE CYBER 2015. He is currently an Associate Editor of IEEE ROBOTICS AND AUTOMATION LETTERS, *International Journal of Robotics and Automation*, *IET Cyber-Systems and Robotics*, and IEEE IROS Conference 2018, 2019, and 2020. He served as a Guest Editor for special issues in IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He was a Program Member of Robotics: Science and Systems 2021.