Unit Dual-Quaternion Parametrisation for Graph SLAM

Jonghyuk Kim, Jiantong Cheng[1] and Hyunchul Shim[2]
Research School of Engineering, The Australian National University
{jonghyuk.kimi}@anu.edu.au
[1]National University of Defense Technology, China
{ cheng.jiantong}@163.com
[2]Korea Advanced Institute of Science and Technology, South Korea
{hcshim}@kaist.ac.kr

Abstract

This paper presents a new parameterisation approach for the graph-based SLAM problem utilising unit dual-quaternion. The rigid-body transformation typically consists of the robot position and rotation, and due to the Lie-group nature of the rotation, a homogeneous transformation matrix (HTM) has been widely used in pose-graph optimizations. In this paper, we investigate the use of unit dual-quaternion (UDQ) for SLAM problem, providing a unified representation of the robot poses with computational and storage benefits. Although UDQ has been widely used in kinematics and navigation (known as Michel Chasles' theorem or Skrew motion), it has not been well utilised in the graph SLAM optimisation. In this work, we re-parameterise the graph SLAM problem using UDQs, focusing on the optimisation performance and the sensitivity to poor initial estimates. Experimental results on public synthetic and real-world datasets show that the proposed approach significantly reduces the computational complexity, whilst retaining the similar accuracies compared to the HTM-based one. With the poor initial estimates, it is also shown that the rotation vector-based perturbation is more stable than the quaternion-based in recovering the error dual-quaternion.

1 Introduction

In Simultaneous Localization and Mapping (SLAM) problem, a robot pose is described as a translation and a rotation, in which the translation is specified by a three-dimensional vector in Cartesian coordinates and the rotation as Euler angles, a unit quaternion or a directional cosine matrix in SO(3) [Kim and Sukkarieh, 2004]. Then, the relative transformation between any two poses can be expressed as a homogeneous transformation matrix (HTM), a 4×4 matrix comprised by corresponding rotation matrix and translation distances.

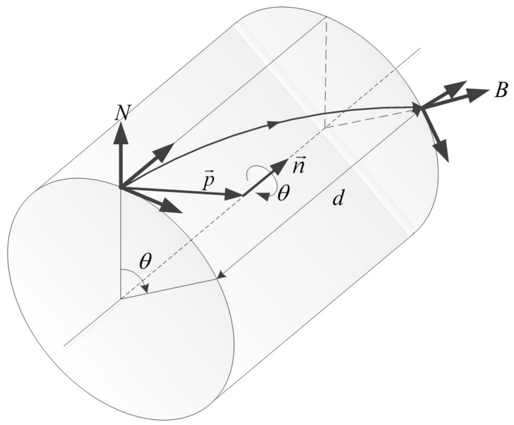


Figure 1: Chasles theorem states that a rigid-body transformation can be performed using a finite rotation θ along a line direction $\vec{n}$ with a pitch displacement d, which is also called screw motion. Unit dual-quaternion is an exponential mapping of the screw vector, effectively capturing the rotation and translation $(\mathbf{R}, \mathbf{t})$

Another popular technique to represent the rigid body transformation is a unit dual-quaternion, which describes a transformation as a twist with only 8 elements, not the aforementioned two separated motions. Compared with the HTM, the dual-quaternion has two interesting advantages in the representation of spatial motions. First, both state variables and their relative transformations can be parameterized as unit dual quaternions (UDQ). Second, the motion compositions can be performed with the UDQ product, which is easy to operate by constructing the dual-quaternion matrix without any non-trivial trigonometric computations.

Motivated by the successful applications of the dual-quaternion in navigation and control areas, we present a new parameterisation approach for the smoothing-based SLAM problem by utilizing the UDQ. After describing relative properties of UDQ, we show that the UDQ is concise and efficient to characterize vertices and edges, and how the error function is reconstructed to avoid the covariance matrix singularities due the two intrinsic constraints in a UDQ. In the optimisation procedure, we prove that the Jacobian matrices in the new approach can be evaluated efficiently without the computation of the matrix derivation with respect to vectors, which is required in HTM. To relieve the unit-norm constraint of unit dual quaternion, minimal representations are suggested to characterise the perturbations in the optimisation procedure. To the best knowledge of authors, there has been no detailed analyse on the performance of different perturbations. Thus, we compare two state-of-the-art over-parameterisations in the SLAM optimisation problem.

The rest of the paper is organized as follows. Section 4 presents the formulation of the graph-based SLAM by using the unit dual-quaternion. In Section 5, the performance of the proposed approach is validated through publicly popular synthetic and practical datasets in 2D and 3D environments, and analyses on the two over-parameterized perturbations are provided based on simulation results. Finally, conclusions are presented in Section 6.

2 Related Work

In the SLAM problem, the smoothing-based approaches consider all measurements in batch to estimate the full trajectory of the robot motion. [Lu and Milios, 1997] is the first in formulating the SLAM problem as a global graph optimisation based on maximum likelihood criterion. Although their approach has a similar formulation as state-of-the-art approaches, the suggested brute-force nonlinear least squares implementation makes their algorithm impractical due to the thousands of nonlinear equations. Subsequently, a large variety of different approaches have been proposed to significantly improve the performances, in terms of efficiency and convergence. For example, [Howard *et al.*, 2001] applied relaxation to localize a robot and build a map. The multi-level relaxation, a variant of Gauss-Seidel relaxation, was proposed in [Frese *et al.*, 2005] to minimize the error at different levels of resolution. [Dellaert and Kaess, 2006] presented square root smoothing and mapping ($\sqrt{SAM}$), in which sparse matrix factorizations are first applied to solve the graphical SLAM problem. [Kaess *et al.*, 2007] proposed incremental SAM (iSAM) for the the real-time implementation of the SLAM optimisation by utilizing a QR-factorization. [Olson *et al.*, 2006] presented an efficient non-linear optimisation approach based on a variant of stochastic gradient descent (SGD), which showed a good stability even with a poor initial guess. A detailed introductory description to the graph-based SLAM problem was provided in [Grisetti *et al.*, 2010] .

All aforementioned optimisation techniques assumed that the state variables span over Euclidean space, which however is not valid for the 3D rotation components in the SLAM problem. To deal with the non-Euclidean parameters, [Hertzberg, 2008] suggested a framework of sparse non-linear least square problems on Manifolds, where state variables are represented with an over-parameterized way, while their increments utilize a minimal representation, namely the translation plus the rotation vector. Recently, [Neuhaus, 2011] applied the manifold framework in a practical SLAM system. [Wagner *et al.*, 2011] published a Manifold Toolkit for Matlab (MTKM) to rapidly construct the manifold-based optimisation systems for multi-sensor calibration and SLAM problems. [Kümmerle *et al.*, 2011] presented a general framework of the manifold-based graphical SLAM optimisation, called g2o, where rotation parts are described by unit quaternions. Furthermore, a simple comparison between the two popular representations of perturbations was provided without further analyses for their different performances.

Contrary to partitioning the rigid body motion into the relative position and orientation, the unit dual-quaternion provides a unified way to describe it. This method shows superior performances in navigation, control, computer vision and so on. [Daniilidis, 1999] firstly applied the UDQ for the camera calibration, in which the camera and motor transformations are represented as screws. Then, their relative transformations can be written with the UDQ product. [Wu *et al.*, 2005] made an in-depth research for the application of the UDQ in strapdown inertial navigation algorithms. Recently, [Wang and Zhu, 2014] made a comparison of the UDQ and the HTM in the robot control, which shows the UDQ outperforms the HTM in the computational efficiency.

3 Unit Dual-Quaternion

3.1 Quaternion

Invented by Hamilton, a *quaternion* is an extension of a complex number to $\mathcal{R}^4$. Without loss of generality, a quaternion is defined as $q = [s, \mathbf{v}^{\mathrm{T}}]^{\mathrm{T}}$[1], where s is a scalar (called *scalar part*), and $\mathbf{v}$ is a 3D vector (called *vector part*). Thus, a vector can be extended to a quaternion with the scalar part being zero, for example, $\mathbf{v}_q = [0, \mathbf{v}]$. Given two quaternions $q_1 = [s_1, \mathbf{v}_1]$ and $q_2 = [s_2, \mathbf{v}_2]$, we have the following operations as

$$q_1 + q_2 = [s_1 + s_2, \mathbf{v}_1 + \mathbf{v}_2] \tag{1}$$

$$\lambda q_1 = [\lambda s_1, \lambda \mathbf{v}_1], \tag{2}$$

[1]For simplicity, the transpose notations for a quaternion are ignored at times, e.g. $q = [s, \mathbf{v}]$.

where $\lambda \in \mathcal{R}$. The multiplication operation of two quaternions is defined as

$$q_1 q_2 = [s_1 s_2 - \mathbf{v}_1^{\mathrm{T}} \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]. \tag{3}$$

In addition, the quaternion product can be simplified as a matrix representation, which is convenient in programming the quaternion products.

$$q_1 q_2 \triangleq [q_1]_+ q_2, \tag{4}$$

where $[q_1]_+$ is the left-product matrix defined as

$$[q_1]_+ = \begin{bmatrix} s_1 & -\mathbf{v}_1^{\mathrm{T}} \\ \mathbf{v}_1 & s_1 I_{3\times3} + [\mathbf{v}_1 \times] \end{bmatrix} \tag{5}$$

with $I_{3\times3}$ being a 3×3 identity matrix, and $[\mathbf{v}\times]$ the skew-symmetric matrix of $\mathbf{v}$. The quaternion norm is defined as $\|q\| = q^{\mathrm{T}} q$. Specially, q is called a unit quaternion if $\|q\| = 1$. The conjugate quaternion of q is defined as $q^* = [s, -\mathbf{v}]$.

Similar to the direction cosine matrix (DCM), a unit quaternion is an efficient tool to represent the rotation. Let a Euler angle vector $A = [\psi, \theta, \phi]_{321}$[2] describe a rotation, the corresponding quaternion can be given by

$$q(A) \triangleq \begin{bmatrix} c_{\phi/2} \cdot c_{\theta/2} \cdot c_{\psi/2} + s_{\phi/2} \cdot s_{\theta/2} \cdot s_{\psi/2} \\ c_{\phi/2} \cdot c_{\theta/2} \cdot s_{\psi/2} - s_{\phi/2} \cdot s_{\theta/2} \cdot c_{\psi/2} \\ c_{\phi/2} \cdot s_{\theta/2} \cdot c_{\psi/2} + s_{\phi/2} \cdot c_{\theta/2} \cdot s_{\psi/2} \\ s_{\phi/2} \cdot c_{\theta/2} \cdot c_{\psi/2} - c_{\phi/2} \cdot s_{\theta/2} \cdot s_{\psi/2} \end{bmatrix}, \tag{6}$$

where $q(A)$ is the mapping function from the Euler angle to the quaternion, with $c_{(\cdot)}$, $s_{(\cdot)}$ representing $\cos(\cdot)$ and $sin(\cdot)$, respectively.

3.2 Dual Number

The *dual number* was proposed by Clifford [Clifford, 1873], and further improved by Study. A dual number is defined as

$$\mathbb{D} = d + \epsilon d', \text{with } \epsilon^2 = 0, \text{but } \epsilon \neq 0, \tag{7}$$

where the scalar components d and d' are called *real part* and *dual part*, respectively. For example, a matrix $\epsilon = [0, x; 0, 0]$ satisfies the above requirement. Specially, we call $\mathbb{D} = d + \epsilon \mathbf{d}'$ a dual vector if $\mathbf{d}'$ is a 3D vector.

The sum and product of two dual numbers $\mathbb{D}_1$ and $\mathbb{D}_2$ are respectively defined as

$$\mathbb{D}_1 + \mathbb{D}_2 = (d_1 + d_2) + \epsilon(d'_1 + d'_2) \tag{8}$$
$$\mathbb{D}_1 \mathbb{D}_2 = d_1 d_2 + \epsilon(d_1 d'_2 + d'_1 d_2). \tag{9}$$

[2]The subscript 321 indicates the rotation order, namely, the first rotation with angle ϕ about z-axis, followed by the rotation with angle θ about y-axis and the rotation with angle ψ about x-axis.

3.3 Unit Dual-Quaternion

A *dual-quaternion* is a special dual number with quaternion components, such as $\mathbb{Q} \triangleq q + \epsilon q' = [\mathbb{S}, \mathbb{V}]$, where $\mathbb{S} = s + \epsilon s$ and $\mathbb{V} = v + \epsilon v'$ are a dual number and a dual vector, respectively. In matrix form, a dual-quaternion can be rewritten as a $8D$ vector, $\mathbb{Q} = [q, q']$. According to Eqs. (3) and (9), the product of two dual quaternions can be given by

$$\begin{aligned} \mathbb{Q}_1 \mathbb{Q}_2 \triangleq & q_1 q_2 + \epsilon(q_1 {q'}_2 + {q'}_1 q_2) \\ = & [\mathbb{Q}_1]_+ \mathbb{Q}_2 \end{aligned} \tag{10}$$

where $[\mathbb{Q}_1]_+$ is the left product matrix of the dual quaternion as

$$[\mathbb{Q}_1]_+ = \begin{bmatrix} [q_1]_+ & 0_{4\times4} \\ [{q'}_1]_+ & [q_1]_+ \end{bmatrix}. \tag{11}$$

Similar to the definition of quaternions, a unit dual-quaternion $\mathbb{Q}$ has

$$\mathbb{Q}\mathbb{Q}^* = [1, 0_{1\times7}]^T \tag{12}$$

where $\mathbb{Q}^* = [q^*, {q'}^*]$ is the conjugate dual-quaternion of $\mathbb{Q}$. Substituting Eq. (10) in (12), we can obtain two intrinsic constraints of a unit dual-quaternion

$$\|q\| = 1, \; q^{\mathrm{T}} q' = 0. \tag{13}$$

It is well known that a unit dual-quaternion has specific meanings for the rigid body transformation, which are nicely presented by the following properties. For brevity, proofs are not provided here. We refer the readers to [Daniilidis, 1999; **?**] for detailed information.

Property 1 *Let A and $\mathbf{t}$ be respectively the rotational angles and translational vector of a rigid body with respect to a global frame, then the rigid-body transformation is equivalent to the following dual-quaternion:*

$$\mathbb{Q}(A, \mathbf{t}) \triangleq [q(A), \; \frac{1}{2}\mathbf{t}q(A)], \tag{14}$$

where, $\mathbb{Q}(\cdot)$ is the mapping function from Cartesian coordinates and Euler angles to the unit dual-quaternion with t being the quaternion form of vector.

Property 2 *Let $\mathbb{Q}_i$ and $\mathbb{Q}_j$ respectively represent the poses of two rigid bodies i and j with respect to the global frame, then the relative pose $\mathbb{Q}_{ij}$ from the rigid body i to j is given by*

$$\mathbb{Q}_{ij} = \mathbb{Q}_i^* \mathbb{Q}_j. \tag{15}$$

4 Graphical SLAM parametrised by Unit Dual-Quaternions

The graph-based SLAM approach constructs a graph with robot poses as its vertices and inter-pose constraints as edges, which are commonly parameterized as $[x, y, z,\ \psi, \theta, \phi]$, with $[x, y, z]$ and $[\psi, \theta, \phi]$ indicating the positing in Cartesian coordinates and Euler angles, respectively. In rigid body kinematics, the robot poses can also be parameterized by Homogeneous Transformation Matrix [Kümmerle *et al.*, 2011; Hertzberg, 2008] or Unit dual-quaternion. Compared to the HTM, the unit dual-quaternion can concisely characterise the pose and the rigid motion only with 8 elements. In particular, the product of two UDQs directly addresses the relative transformation between two rigid bodies, without the cumbersome transformations between the over-parameterized representations and rotation matrices. Thus, the UDQ can be a more efficient parameterisation method in Graphical SLAM.

Figure 2 shows an example of a directed graphical SLAM with the classical pose representation of vertices and edges 2(a) and with the unit dual-quaternion parametrization shown in Figure 2(b). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a directed graph of SLAM, where $\mathcal{V} = \{x_i\}$ and $\mathcal{E} = \{z_{ij}\}$ are the sets of vertices and edges. The edge $z_{ij} \triangleq (i, j)$ denotes the constraint from x_i to x_j. In particular, if $j = i+1$, z_{ij} denotes a sequential motion constraint obtained from the odometry measurement, otherwise it indicates a loop closure constraint, which can be obtained from, for example, scan matching.

Using UDQs parameterisation, the pose vertices are represented as $\mathcal{V} = \{\mathbb{Q}_i\}$ and the measurement edges $\mathcal{E} = \{\mathbb{Q}_{ij}\}$ with related information matrix $\{\Lambda_{ij}\}$.

4.1 Error Function

Under the Euclidean space assumption, the measurement error corresponding to z_{ij} is generally defined as

$$e_{ij} \triangleq f_{ij}(x_i, x_j) - z_{ij}, \tag{16}$$

where $f_{ij}(\cdot)$ is the nonlinear measurement function from x_i to x_j, and $e_{ij} \sim \mathcal{N}(0, \Omega_{ij}^{-1})$ is defined in $SE(3)$. Ω_{ij} is the known information matrix dependent on the sensors applied in the SLAM system.

Since the uncertainties are expressed for the translations and Euler angles, it is required to transform them to the uncertainties of UDQs. Given the estimation of pose UDQ, the error UDQ according to Property 2 can be calculated as

$$\begin{aligned}\mathbb{E}_{ij}(\mathbb{Q}_i, \mathbb{Q}_j) &= \mathbb{Q}_{ij}^* \mathbb{Q}_i^* \mathbb{Q}_j \\ &= [\mathbb{Q}_{ij}^*]_+ [\mathbb{Q}_i^*]_+ \mathbb{Q}_j. \end{aligned}\tag{17}$$

According to Property 1, the transformed covariance of the error UDQ, Γ_{ij}, can then be obtained by utilising a Jacobian matrix

$$\Gamma_{ij} \triangleq F_{ij} \Omega_{ij}^{-1} F_{ij}^T, \quad \text{with } F_{ij} = \frac{\partial \mathbb{E}_{ij}}{\partial e_{ij}}. \tag{18}$$

Note that the UDQ matrix is specified by its elements directly without any other computations. Although E_{ij} is a nonlinear function of $\mathbb{Q}$, it shows linear relationships with $\mathbb{Q}_i$ or $\mathbb{Q}_j$. Thus, the Jacobian matrix of $\mathbb{E}_{ij}$ with respect to $\mathbb{Q}$ can be easily evaluated. See Appendix A for the detailed structure of F_{ij}.

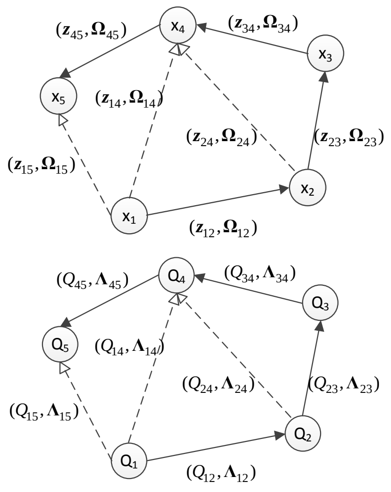


Figure 2: An example diagram of pose-graph SLAM parameterized by Cartesian coordinates plus Euler angles and their information matrix $(z_{ij},\ \mathbf{\Omega}_{ij})$ (top) and unit dual-quaternions with information matrix $(\mathbb{Q}_{ij},\ \mathbf{\Lambda}_{ij})$ (bottom), respectively. Solid lines denote sequential measurements, whilst dashed lines represent loop-closure constraints.

It should be noted that due to the two constraints in UDQ, E_{ij} is over-parameterized and Γ_{ij} is not invertible. To avoid the singularity of Γ_{ij}, we evaluate only 6 elements of $\mathbb{E}_{ij}$ in the optimisation procedure

$$\mathbb{E}_{ij} = (\mathbb{E}_{ij})_{[2:4,6:8]} \tag{19}$$

$$\Lambda_{ij} = ((\Gamma_{ij})_{[2:4,6:8]})^{-1}, \tag{20}$$

where the operator $(\cdot)_{[2:4,6:8]}$ picks up the rows $[2:4, 6:8]$ of a vector, and corresponding rows and columns of a matrix. Λ_{ij} is the information matrix of $\mathbb{E}_{ij}$.

Thus, $\mathbb{E}_{ij} \sim \mathcal{N}(0, \Lambda_{ij}^{-1})$. By using the two constraint equations in Eq. (13), the full error UDQ can be easily recovered.

4.2 Optimisation on Unit Dual Quaternions

Similar to [Cheng *et al.*, 2015], the solution to a pose-graph SLAM problem is equivalent to minimising the following error function in UDQs to obtain the optimal pose configuration.

$$\underset{\{\mathbb{Q}\}}{\text{argmin}} \sum_{(i,j)\in\mathcal{E}} \mathbb{E}_{ij}^T \Lambda_{ij} \mathbb{E}_{ij}. \tag{21}$$

Due to the nonlinearity of $\mathbb{E}_{ij}$ with respect to $\mathbb{Q}$ in Eq. (17), the first-order Taylor expansion around its estimation $\hat{\mathbb{Q}}$ is commonly applied to approximate the error function. The perturbation of $\mathbb{Q}_i$ can be directly performed on UDQs which, however, has added complexities due to the unit constraint. In this work, we only over-parameterize the rotational parts for the perturbation, whilst the translations are still described in Cartesian coordinates.

Let δx_i be the minimal representation of the error $\delta\mathbb{Q}_i$. In this work we adopted two state-of-the-art representations: a) Cartesian + rotation vector and b) Cartesian + quaternion. Then the correction can be performed by first converting the δx_i into a corresponding $\delta\mathbb{Q}_i$,

$$\mathbb{Q}_i \triangleq \mathbb{Q}_i\delta\mathbb{Q}_i = \mathbb{Q}_i\delta\mathbb{Q}_i(\delta X), \tag{22}$$

with $\delta X = \{\delta x_i\}$. Please refer to Appendix B for the details of two representations and their related Jacobians.

The first-order approximation of the error dual quaternion $\mathbb{E}_{ij}(\mathbb{Q})$ in Eq. 21 is

$$\mathbb{E}_{ij}(\delta X) \simeq \hat{\mathbb{E}}_{ij} + J_{ij}\cdot\delta X \tag{23}$$

where J_{ij} is the Jacobian matrix of $\mathbb{E}_{ij}(\mathbb{Q})_{[2:4,6:8]}$ computed in $\mathbb{Q}$. Note that the error UDQ, $\mathbb{E}_{ij}$, depends only two related vertices $\mathbb{Q}_i$ and $\mathbb{Q}_j$, J_{ij} thus has the following sparse form

$$J_{ij} = [\cdots, \underbrace{A_{ij}}_{\text{node } i}, \cdots, \underbrace{B_{ij}}_{\text{node } j}, \cdots], \tag{24}$$

where $\cdots$ indicates zero elements of J_{ij}, the blocks A_{ij} and B_{ij} respectively denote these corresponding derivatives of $\mathbb{E}_{ij}(\mathbb{Q})$ with respect to δx_i and δx_j, and can be computed analytically through the chain rule as

$$A_{ij} = \left.\frac{\partial\mathbb{E}_{ij}(\delta X)}{\partial\delta x_i}\right|_{\delta X=0} = M_r[\mathbb{Q}_j]_-I_c[\mathbb{Q}_i]_+G \tag{25}$$

$$B_{ij} = \left.\frac{\partial\mathbb{E}_{ij}(\delta X)}{\partial\delta x_j}\right|_{\delta X=0} = M_r[\mathbb{Q}_i^*]_+[\mathbb{Q}_j]_+G, \tag{26}$$

where $M_r = ([\mathbb{Q}_{ij}^*]_+)_{[2:4,6:8]}$, and G is the Jacobian matrix of $\mathbb{Q}(\cdot)$ at $\delta X = 0$, and $I_c = \mathrm{diag}([1, -\mathrm{ones}(1,3), 1, -\mathrm{ones}(1,3)])$ so that $\mathbb{Q}^* = I_c\mathbb{Q}$.

It can be seen fromAppendix B that G is a constant matrix no matter which over-parameterisation method is used. dual-quaternion matrices in J_{ij} have been evaluated in computing measurement errors. Thus, we can efficiently compute the Jacobian matrices through the multiplication of these matrices. In contrast, the analytical solution of the Jacobian matrices in HTM-based approach requires the derivatives of rotation matrices with respect to rotation vectors. Consequently a common approach in HTM-based is to numerically compute the Jacobian matrices, resulting in a high computational complexity.

Substituting the above approximation into Eq. 21 gives

$$\begin{aligned}\mathbb{E}_{ij}^T\Lambda_{ij}\mathbb{E}_{ij} &\simeq (\hat{\mathbb{E}}_{ij} + J_{ij}\delta X)^{\mathrm{T}}\Lambda_{ij}(\hat{\mathbb{E}}_{ij} + J_{ij}\delta X) \\ &= \underbrace{\hat{\mathbb{E}}_{ij}^{\mathrm{T}}\hat{\mathbb{E}}_{ij}}_{c_{ij}} + 2\underbrace{\hat{\mathbb{E}}_{ij}^{\mathrm{T}}\Lambda_{ij}J_{ij}}_{b_{ij}^{\mathrm{T}}}\delta X + \delta X^{\mathrm{T}}\underbrace{J_{ij}^{\mathrm{T}}\Lambda_{ij}J_{ij}}_{H_{ij}}\delta X,\end{aligned} \tag{27}$$

where $c_{ij} = \hat{\mathbb{E}}_{ij}^{\mathrm{T}}\hat{\mathbb{E}}_{ij}$, $b_{ij} = J_{ij}^{\mathrm{T}}\Lambda_{ij}\hat{\mathbb{E}}_{ij}$ and $H_{ij} = J_{ij}^{\mathrm{T}}\Lambda_{ij}J_{ij}$. Substituting Eq. (24) in them, we obtain

$$b_{ij} = \begin{bmatrix} \vdots \\ A_{ij}^{\mathrm{T}}\Lambda_{ij}\hat{\mathbb{E}}_{ij} \\ \vdots \\ B_{ij}^{\mathrm{T}}\Lambda_{ij}\hat{\mathbb{E}}_{ij} \\ \vdots \end{bmatrix} \tag{28}$$

$$H_{ij} = \begin{bmatrix} \ddots & & & & \\ & A_{ij}^{\mathrm{T}}\Lambda_{ij}A_{ij} & \cdots & A_{ij}^{\mathrm{T}}\Lambda_{ij}B_{ij} & \\ & \vdots & \ddots & \vdots & \\ & B_{ij}^{\mathrm{T}}\Lambda_{ij}A_{ij} & \cdots & B_{ij}^{\mathrm{T}}\Lambda_{ij}B_{ij} & \\ & & & & \ddots \end{bmatrix} \tag{29}$$

Since the origin node (node 0) is fixed, we do not need to augment it in the state vector. Subsequently, $A_{0j} = 0$ and its blocks in b_{ij} and H_{ij} are omitted.

Now, the total error function over the network in Eq. (21) can be rewritten as

$$\begin{aligned}&\operatorname{argmin}\sum_{(i,j)\in\mathcal{E}}(c_{ij} + 2b_{ij}^{\mathrm{T}}\delta X + \delta X^{\mathrm{T}}H_{ij}\delta X) \\ &\quad = \operatorname{argmin}\left\{c + 2b^{\mathrm{T}}\delta X + \delta X^{\mathrm{T}}H\delta X\right\},\end{aligned} \tag{30}$$

where $c = \sum_{(i,j)\in\mathcal{E}} c_{ij}$, $b = \sum_{(i,j)\in\mathcal{E}} b_{ij}$, and $H = \sum_{(i,j)\in\mathcal{E}} H_{ij}$. The solution to minimizing the error function is then obtained by setting its first-order derivative with respect to δX to zero, giving

$$H\delta X = -b. \tag{31}$$

Due to the non-zero blocks of H_{ij} depending only on two vertices connected by the constraint, H is, in fact, the adjacency matrix of the graph. Thus, the number of non-zero blocks in H is proportional to the number of constraints in the pose graph, resulting in a sparse structure. Since each pose has a corresponding sequential measurement, H is a full rank matrix. δX can be efficiently solved by the sparse matrix inversion in Matlab as $\delta X = -H\backslash b$. Then the state unit-dual quaternions are updated using δX in Eq. (22).

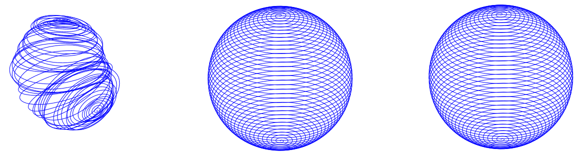
5 Experimental Results

In this section, we perform a series of experiments to verify the performances of the proposed approach by using publicly available datasets in $2D$ and $3D$ environments. Although four different datasets are used, namely $City10000$, $ManhattanOlson3500$, Intel Research Lab and Parking Garage datasets, only the Parking Garage results will be shown in this paper due to the limited space. We first compare the performance of the aforementioned two minimal-representation for the quaternion purtabation: that is a) quaternion-based and b) rotation vector-based. Simulated $Sphere2500$ datasets with different noise levels are used together with the open-source g2o solver [Kümmerle *et al.*, 2011].

5.1 Comparision of Two Perturbation Methods

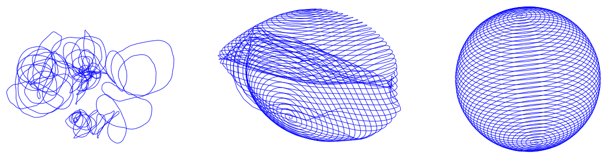
As described in Section 4.2, the unit quaternion and the angle-axis representations are the most popular methods in representing 3D rotations without singularities. In the UDQ perturbation step, either method can be used to compute the error UDQ. In the section, we evaluate their sensitivities with respect to initial errors. Figures 3 and 4 visualize the convergence results of two perturbation methods with different initial errors. For clarity, the coordinate axes are not shown in figures. For small initial error, both methods converge quite quickly no matter which perturbation method is adopted. The final estimation errors are also nearly the same. However, the two methods show significant differences with the large initial error. The solution based on the rotation vector representation converges after 15 iterations, whilst, the quaternion-based method converges much slowly, requiring more than twice number of iterations.

The different convergence properties in the two over-parameterisations are mainly due to the conversions from their minimal representations to corresponding full representations. That is, the vector part of the quaternion is used as its minimal representation, and then the full quaternion is recovered using Eq. (40). However, two potential problems can happen during the conversion: the sign ambiguity of the scalar part in the square root operation and the singularity when the norm of the estimated vector part is larger than 1. In the literature, the perturbations around the estimate are usually assumed to be small. This assumption is reasonable when the initial guess is reasonably good. To address this problem, Kümmerle *et al.* [Kümmerle *et al.*, 2011] correct the current angle increments to be zeros in g2o, namely $q = [1, 0, 0, 0]$ under $\|v\| > 1$. However, both g2o and our algorithms with this correction cannot obtain correct estimates for the large initial error as in the second experimental dataset. Thus, we improve the conversion by normalizing the vector part, whilst setting the scalar



(a) Initial guess (b) Quaternion (c) Rotation vector

Figure 3: Convergence with low-level of initial error for two perturbations (quaternion- and rotation vector-based) using Sphere2500 dataset. (a) illustrates the initial guess of the full trajectory with low noise. (b) and (c) show the results after 5th iteration of the quaternion-based and rotation vector-based method, respectively, showing similar performance with low initial error.



(a) Initial guess (b) Quaternion 19th iteration (c) Quaternion 43th iteration

Figure 4: Convergence with high-level of initial error. (a) shows the initial guess with high-level of noise. (b) and (c) show the quaternion-based results after 19th and 43th iteration, respectively. Rotation vector method shows similar convergence after 15th iteration (not shown here for breivity).

part to zero, $q = [0,\ v/\|v\|]$ when $\|v\| > 1$ in our experiments. Since it is hard to specified the sign of the scalar part, we pick up the positive one in our algorithm as in g2o. This approximation might not be correct but will be fixed in the next iterations. Contrarily, the transformation from the minimal representation of the rotation vector to the quaternion is unique and continuous as shown in Eq. (46). Thus, the over-parameterized method based on the rotation vector becomes more numerically stable.

5.2 Comparison with HTM

In this section, we compare our approach with the HTM in aspects of accuracies and runtime. The main difference between UDQ and HTM in SLAM is the representation of the rigid body transformation. The HTM-based optimisation algorithm can be found in MTKM [Wagner *et al.*, 2011; Hertzberg, 2008], which however applies a series of classes to store the computation results, resulting in serious time cost. For the fair comparison, we implement the optimisation algorithm based on the HTM with the same structure to our approach. All implementations are performed in Matlab on a computer with dual Pentium E6600 processors at 3.06GHz and 2GB of RAM.

Absolute trajectory error and relative pose error are two popular performance metrics to evaluate the result

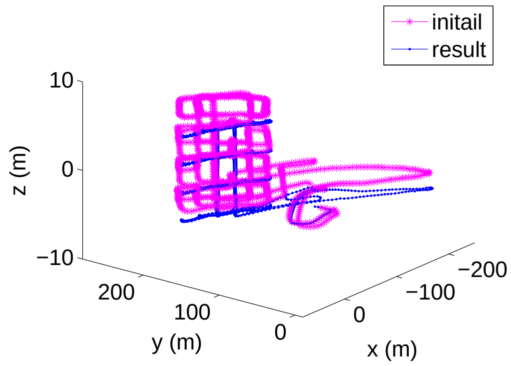
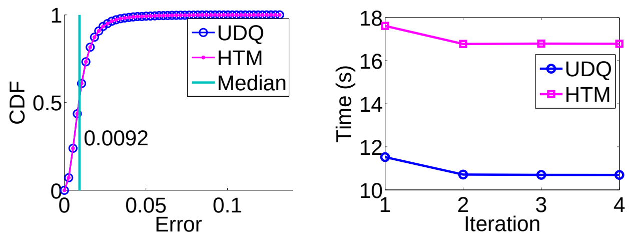


Figure 5: Optimisation results of the UDQ-based algorithm on Parking Garage dataset, with star-lines representing the initial guess obtained from the odometry measurements.



(a) Accuracy in CDF (b) Computational Time

Figure 6: (a) Cumulative Distribution Functions (CDF) of the RPE e_{ij}. Here, solid lines indicate the medians. The vertical axis in illustrates the probability of the RPE less than or equal to the given error in the horizonal axis. (b) Time cost by each iteration of the UDQ-based and the HTM-based algorithms.

of a SLAM algorithm. However, [Kümmerle *et al.*, 2009] shows that ATE is suboptimal due to a single error in a pose resulting in errors in its following poses. In addition, the ground truth is usually not available in practical experiments. Conversely, the RPE measures the differences between the estimated and true motions, without the requirement of the ground truth. Thus, we evaluate the following experimental results by the RPE. The UDQ-based optimisation results show almost the same accuracy to the HTM-based method for the four datasets, which is expected. Figure 6(a) depicts the Cumulative Distribution Functions (CDF) [Cheng *et al.*, 2015] of the RPE for the Parking Garage dataset, with less than 0.05 RPE in both approaches.

The computational time required for one iteration of both approaches are shown in Figure 6(b). It can be clearly seen that the UDQ-based method is roughly 1.5 times faster than the HTM-based. In computing the Jacobian matrices, numerical method was used rather than analytical one, in which case, the time further reduced from $11sec$ to $4sec$. From these results, it is clear that the UDQ-method can achieve substantial improvement in the computational efficiency, whilst maintaining the graph SLAM accuracy.

6 Conclusions

In this paper, we re-parameterized the graphical SLAM in terms of unit dual-quaternions, which provided a unified representation of robot poses with computationally efficiency. The vertices and edges are represented with UDQs and the relative transformation between two vertices are expressed as a UDQ matrix. We formulated the measurement errors using the UDQ product, which leads to a linear form with respect to each pose UDQ. Consequently, the Jacobians of the error function could be computed efficiently as a product of corresponding UDQ matrices. Since the linearized error function has the same structure as state-of-the-art approaches, the Gauss-Newton was directly applied to solve this optimisation problem. In addition, we showed that, in computing error UDQs, the angle-axis method showed improved convergence than the quaternion-based under large initial errors, which is related to the numerical instability of quaternion in recovering from the minimal representation. The future work is to further investigate the convergence property of the quaternion and apply the method for estimating the trajectory and map from an autonomous car.

A Jacobian $F(e_{ij})$

In the follows, we provide the detailed structure of F_{ij}, the Jacobian matrix of $\mathbb{E}_{ij}$ with respect to e_{ij}.

Let the pose error $e_{ij} = [\mathbf{t}_{ij}, A_{ij}]$ be defined as the errors in Cartesian position and Euler angles. Then the equivalent error in unit dual-quaternion $E_{ij}(e_{ij})$ in Eq. (14) becomes $E_{ij} \triangleq [q, q'] = [q(A_{ij}),\ 0.5\mathbf{t}_{ij}q(A_{ij})]$. Then F_{ij} becomes

$$F_{ij} = \frac{\partial E_{ij}}{\partial e_{ij}} = \begin{bmatrix} \frac{\partial q(A_{ij})}{\partial t_{ij}} & \frac{\partial q(A_{ij})}{\partial A_{ij}} \\ \frac{\partial (0.5\mathbf{t}_{ij}q(A_{ij})}{\partial t_{ij}} & \frac{\partial (0.5\mathbf{t}_{ij}q(A_{ij})}{\partial A_{ij}} \end{bmatrix} \quad (32)$$

Note that $e_{ij} \sim N(0, \Omega_{ij}^{-1})$, we have $q(A_{ij}) = [1,0,0,0]$ and $[q(A_{ij})]_- = I_{4\times4}$ at $e_{ij} = 0$. Substituting Eq. (6) into the above equation, we obtain

$$\frac{\partial q(A_{ij})}{\partial t_{ij}} = 0_{4\times3} \quad (33)$$

$$\frac{\partial q(A_{ij})}{\partial A_{ij}} = \begin{bmatrix} 0_{1\times3} \\ 0.5I_{3\times3} \end{bmatrix} \quad (34)$$

$$\frac{\partial (0.5\mathbf{t}_{ij}q(A_{ij}))}{\partial t_{ij}} = \begin{bmatrix} 0_{1\times3} \\ 0.5I_{3\times3} \end{bmatrix} \quad (35)$$

$$\frac{\partial(0.5\mathbf{t}_{ij}q(A_{ij}))}{\partial A_{ij}} = 0_{4\times 3} \tag{36}$$

Finally, F_{ij} is simply given as

$$F_{ij} = \begin{bmatrix} 0_{1\times 3} & 0_{1\times 3} \\ 0_{3\times 3} & 0.5I_{3\times 3} \\ 0_{1\times 3} & 0_{1\times 3} \\ 0.5I_{3\times 3} & 0_{3\times 3} \end{bmatrix}. \tag{37}$$

Using this Jacobian matrix, the uncertainty (or information) matrix in traditional pose graph can be transformed into that of unit dual-quaternions.

B Minimal Representations of Perturbation

Given the minimal representation of the pose error, δx_{ij}, the Jacobian matrix G can be evaluated

$$\delta\mathbb{Q}_{ij} = \left.\frac{\partial\mathbb{Q}_{ij}(x_{ij})}{\partial x_{ij}}\right|_{x=0} \delta x_{ij}. \tag{38}$$

B.1 Quaternion

In this method, the minimal representation of perturbations is based on position error and quaternion error

$$\delta x = [\delta t_x, \delta t_y, \delta t_z, \delta q_1, \delta q_2, \delta q_2] \triangleq [\delta\mathbf{t}, \delta\mathbf{v}] \tag{39}$$

The unit dual-quaternion of perturbations $\delta\mathbb{Q}_i \triangleq [\delta q_i, \delta q'_i] = g(\delta x_i)$ can be computed as

$$\delta q_i = \left[\sqrt{1-\delta\mathbf{v}_i^{\mathrm{T}}\delta\mathbf{v}_i}, \quad \delta\mathbf{v}_i\right] \triangleq [\delta s_i, \delta\mathbf{v}_i] \tag{40}$$

$$\delta q'_i = 0.5\ \delta\mathbf{t}_i\delta q_i \tag{41}$$

Given $\delta x_i = 0$, the Jacobian matrix G is given by

$$G_{8\times 6} = \begin{bmatrix} 0_{1\times 3} & \frac{-\delta\mathbf{v}_i^{\mathrm{T}}}{\sqrt{1-\delta\mathbf{v}_i^{\mathrm{T}}\delta\mathbf{v}_i}} \\ 0_{1\times 3} & I_{3\times 3} \\ 0.5\delta v_i & 0_{1\times 3} \\ 0.5(\delta s_i I_{3\times 3} + [\delta v_i\times]) & 0_{1\times 3} \end{bmatrix}_{\delta x_i=0} \tag{42}$$

$$= \begin{bmatrix} 0_{1\times 3} & 0_{1\times 3} \\ 0_{3\times 3} & I_{3\times 3} \\ 0_{1\times 3} & 0_{1\times 3} \\ 0.5I_{3\times 3} & 0_{3\times 3} \end{bmatrix} \tag{43}$$

B.2 Rotation Vector

The rotation vector (or axis-angle) representation, $\mathbf{r} = [\theta, \hat{\mathbf{n}}] = [r_x, r_y, r_z]$, parameterizes a rotation by a *unit* vector $\hat{\mathbf{n}}$ indicating the rotation axis and an angle θ representing the rotation magnitude about the axis. Then, the minimal representation of perturbations using the rotation vector is given by

$$\delta x = [\delta t_x, \delta t_y, \delta t_z, \delta r_x, \delta r_y, \delta r_z] \triangleq [\delta\mathbf{t}, \delta\mathbf{r}] \tag{44}$$

The corresponding unit dual quaternion error is

$$\delta\mathbb{Q}_i = [\delta q_i, \delta q'_i] = [\delta q_i, 0.5\delta\mathbf{t}_i\delta q_i], \tag{45}$$

where

$$\delta q_i = \begin{cases} [1,0,0,0], & \text{if } \|\delta\mathbf{r}_i\| = 0 \\ \left[\cos\left(\frac{\|\delta\mathbf{r}_i\|}{2}\right), \frac{\delta\mathbf{r}_i}{\|\delta\mathbf{r}_i\|}\sin\left(\frac{\|\delta\mathbf{r}_i\|}{2}\right)\right] & \text{otherwise} \end{cases} \tag{46}$$

with its derivative evaluated as

$$\frac{\partial\delta q_i}{\partial\delta\mathbf{r}_i} = \begin{bmatrix} -\frac{\delta\mathbf{r}_i^{\mathrm{T}}}{2\|\delta\mathbf{r}_i\|}\sin\left(\frac{\|\delta\mathbf{r}_i\|}{2}\right) \\ \cos\left(\frac{\|\delta\mathbf{r}_i\|}{2}\right)\frac{\delta\mathbf{r}_i\delta\mathbf{r}_i^{\mathrm{T}}}{2\delta\mathbf{r}_i^{\mathrm{T}}\delta\mathbf{r}_i} + \sin\left(\frac{\|\delta\mathbf{r}_i\|}{2}\right)\frac{\delta\mathbf{r}_i^{\mathrm{T}}\delta\mathbf{r}_i I - \delta\mathbf{r}_i\delta\mathbf{r}_i^{\mathrm{T}}}{\delta\mathbf{r}_i^{\mathrm{T}}\delta\mathbf{r}_i} \end{bmatrix} \tag{47}$$

Specially, we have the left derivative and right derivative as

$$\left.\frac{\partial\delta q_i}{\partial\delta\mathbf{r}_i}\right|_{\delta\mathbf{r}_i\to 0^-} = \left.\frac{\partial\delta q_i}{\partial\delta\mathbf{r}_i}\right|_{\delta\mathbf{r}_i\to 0^+} = \begin{bmatrix} 0_{1\times 3} \\ 0.5I_{3\times 3} \end{bmatrix} \tag{48}$$

Then, the Jacobian matrix G can be obtained as

$$G_{8\times 6} = \begin{bmatrix} 0_{1\times 3} & 0_{1\times 3} \\ 0_{3\times 3} & 0.5I_{3\times 3} \\ 0_{1\times 3} & 0_{1\times 3} \\ 0.5I_{3\times 3} & 0_{3\times 3} \end{bmatrix} \tag{49}$$

References

[Cheng *et al.*, 2015] J. Cheng, J. Kim, J. Shao, and W. Zhang. Robust linear pose graph-based slam. *Robotics and Autonomous System*, 72:71–82, 2015.

[Clifford, 1873] W. K. Clifford. Preliminary sketch of bi-quaternions. *Proceeding of London Mathematical Socity*, 4:381–395, 1873.

[Daniilidis, 1999] K. Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, 18, 1999.

[Dellaert and Kaess, 2006] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[Frese *et al.*, 2005] Udo Frese, Per Larsson, and Tom Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, 2005.

[Grisetti *et al.*, 2010] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[Hertzberg, 2008] Christoph Hertzberg. *A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds.* PhD thesis, 2008.

[Howard *et al.*, 2001] Andrew Howard, Maja J Mataric, and Gaurav Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1055–1060, Maui, Hawaii, 2001.

[Kaess *et al.*, 2007] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Fast incremental smoothing and mapping with efficient data association. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1670–1677, Roma, Italy, 2007.

[Kim and Sukkarieh, 2004] J. Kim and S. Sukkarieh. Autonomous airborne navigation in unknown terrain environments. *IEEE Transactions on Aerospace and Electronics Systems*, 40(3):1031–1045, 2004.

[Kümmerle *et al.*, 2009] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[Kümmerle *et al.*, 2011] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE Internal Conference on Robotics and Automation (ICRA)*, pages 3607 – 3613, Shanghai, May 2011.

[Lu and Milios, 1997] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[Neuhaus, 2011] F. Neuhaus. *A full 2d/3d graphslam system for globally consistent map- ping based on manifolds.* PhD thesis, 2011.

[Olson *et al.*, 2006] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269, Orlando, 2006.

[Wagner *et al.*, 2011] R. Wagner, O. Birbach, and U. Frese. Rapid development of manifold-based graph optimization systems for multi-sensor calibration and slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[Wang and Zhu, 2014] Xiangke Wang and Huayong Zhu. On the comparisons of unit dual quaternion and homogeneous transformation matrix. *Advances in Applied Clifford Algebras*, 24(1):213–229, 2014.

[Wu *et al.*, 2005] Yuanxin Wu Wu, Xiaoping Hu Hu, Dewen Hu Hu, Tao Li, and Junxiang Lian. Strapdown inertial navigation system algorithms based on dual quaternions. *IEEE Transactions on Aerospace and Electronic Systems*, 41:110–132, 2005.