Pose Estimation with Dual Quaternions and Iterative Closest Point

Aksel Sveier[1], Torstein A. Myhre[1] and Olav Egeland[1]

Abstract— **This paper presents a method for pose estimation of a rigid body using unit dual quaternions where pose measurements from point clouds are filtered with a multiplicative extended Kalman filter (MEKF). The point clouds come from a 3D camera fixed to the moving rigid body, and then consecutive point clouds are aligned with the Iterative Closest Point (ICP) algorithm to obtain pose measurements. The unit constraint of the dual quaternion is ensured in the filtering process with the Dual Quaternion MEKF (DQ-MEKF), where the measurement updates are performed using the dual quaternion product so that the result is a unit dual quaternion. In addition, we use the Cayley transform for the discrete time propagation of the DQ-MEKF estimate, eliminating the need for normalization and projection of the resulting unit dual quaternion. The ICP algorithm is initialized with the time propagated state of the filter to give faster and more accurate pose measurements. To further improve the accuracy of the initialization, angular velocity measurements from a gyroscope fixed to the camera are included in the filter. The proposed method has been tested in experiments using a Kinect v2 3D camera mounted rigidly on a KUKA KR6 robotic arm, while a customized ICP algorithm was successfully implemented on a Graphical Processing Unit (GPU) system.**

I. INTRODUCTION

Relative pose estimation of systems with six degrees-of-freedom is important in relative navigation [9], 3D mapping [11], and robotics [7], [3]. The availability of commodity real-time 3D cameras has opened up for new approaches for measuring and observing relative position and attitude. 3D cameras sample the observed scene in a 3D image, which can be represented as a set of points called a point cloud. Pose information from point clouds can be obtained by registration algorithms such as the Iterative Closest Point (ICP) [2]. The pose information can describe the movement of the camera, or the movement of an object observed by the camera. In this paper we focus on the former case, but the same methods are applicable for the latter. By treating registration results as pose measurements, these can be filtered in agreement with the dynamics and process noise of the observed system.

The ICP algorithm is performed in two steps; first point-to-point nearest neighbour (NN) correspondences between two point clouds are obtained, then the transformation that minimize an error metric between the correspondences is found. It is assumed that the two point clouds are equal in the sense that there exist a common rigid displacement that will exactly match all the points in one point cloud to the other. In [20] it was suggested to use a modified ICP algorithm for aligning point clouds from 3D sensors. This is challenging due to sensor noise and difference in viewpoint between consecutive point clouds. They suggest two constraint on the NN correspondences; in the first constraint corresponding pairs of points are discarded when they are too far apart to filter out false correspondences and reduce the effect of noisy outliers. In the second constraint a corresponding pair is discarded when either point is on a mesh boundary to reduce alignment error due to partly overlapping point clouds.

In [1], ICP was used in combination with an adaptive Kalman filter (AKF) to obtain the relative pose between a space station and a spacecraft by aligning a laser scan of the spacecraft with a CAD model. Fault detection for the ICP pose estimates was implemented using measurements from an Inertial Measurement Unit (IMU). In [10], the ICP measurements from 3D data were used in combination with an Invariant Extended Kalman Filter (IEKF) to create 3D maps. It was shown that the filter produced better results than ICP alone.

The measurement update for estimating elements on the $SE(3)$ and $SO(3)$ group is nonlinear. Nonlinear filters such as the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) have been applied to handle this nonlinearity [12]. The Quaternion Multiplicative EKF (Q-MEKF) [13] has a multiplicative measurement update based on the quaternion product to produce a resulting unit quaternion, and only the vector part of the error quaternion is used in the Kalman filter equations, reducing the dimension of the covariance matrix. This filter has been applied to successfully estimate attitude of NASA spacecrafts [5].

Unit dual quaternions can be used to represent both attitude and position. The recently proposed Dual Q-MEKF (DQ-MEKF) [9] is an extension of the Q-MEKF method where the pose is estimated using a unit dual quaternion. A multiplicative update is performed with the dual quaternion product, and only the vector part is used in the filter equations to reduce the dimension of the covariance matrix. The time propagation of the dual quaternion in the filter is followed by a normalization and projection operation to ensure that the result is a unit dual quaternion. In [6] an unscented Kalman filter for pose estimation is developed. Here, the dual quaternion dynamics is propagated with an exponential function to avoid violation of the unit constraint.

In this paper, we present a method for integrating the DQ-MEKF filter with the ICP algorithm. As an extension of the results in [9], [6], the Cayley transform [19] is used as an approximation of the exponential function in the time propagation of the filter. This gives a unit dual quaternion, so that normalization and projection is not needed. The registration process of the ICP algorithm is initialized with

[1]The authors are with the Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway {aksel.sveier, torstein.a.myhre, olav.egeland}@ntnu.no

the time propagated state of the filter, which gives faster and more accurate ICP results [1], [10]. Furthermore, we validated the integrated filter in an experiment where a 3D camera was mounted on a robotic arm. The integrated filter estimated the path of the robot by observing the rigid environment without requiring prior knowledge about geometry or features of the surroundings. The experiment showed a significant improvement in accuracy when the ICP algorithm was initialized with the time propagated state of the filter. Further improvement in accuracy was achieved by including angular velocity measurements from a gyroscope fixed to the camera. Moreover, we implemented the constraints on the NN correspondences from [20] in our ICP algorithm and developed a method for determining mesh boundaries in a point cloud.

The paper is organized as follows: Sect. II present notation and preliminary results of quaternions, dual quaternions, the Cayley transform and the DQ-MEKF. In Sect. II-E we present the discrete time propagation of the filter. The ICP algorithm is presented in Sect. III. Details regarding the NN search is presented in Sect. III-A, detection of mesh boundaries is presented in Sect. III-B and our implementation of ICP and is presented in Sect. III-C. Furthermore, the integration of ICP and the DQ-MEKF is presented in Sect. IV, calibration of camera and gyroscope is presented in Sect. V and the experimental results are presented in Sect. VI. Finally, the paper is concluded in Sect. VII.

II. Pose Estimation with Dual Quaternions

A. Quaternions

A quaternion $q = q_0 + \bar{q}$ is written as the sum of a scalar q_0 and a vector $\bar{q} = [q_1\ q_2\ q_3]^\top$ [8]. The quaternion q can also be represented as the vector $[q] = [q_0\ q_1\ q_2\ q_3]^\top$. Addition of two quaternions $a = a_0 + \bar{a}$ and $b = b_0 + \bar{b}$ is defined component-wise and the quaternion product is given by

$$a \otimes b = (a_0 b_0 - \bar{a} \cdot \bar{b}) + (a_0 \bar{b} + b_0 \bar{a} + \bar{a} \times \bar{b}). \tag{1}$$

The conjugate of a quaternion q is denoted by $q^* = q_0 - \bar{q}$ and the magnitude of q is

$$\|q\|^2 = q \otimes q^* = [q]^\top [q] = q_0^2 + \bar{q} \cdot \bar{q}. \tag{2}$$

The inverse of a quaternion is $q^{-1} = q^*/\|q\|^2$.

A vector $\bar{v} \in \mathbb{R}^3$ is written in quaternion form as $v = 0 + \bar{v}$ with zero scalar part. The skew-symmetric matrix of a vector is denoted as $\bar{v}^\times$. A quaternion which satisfies $q \otimes q^* = 1$ is called a unit quaternion. A unit quaternion can be used to represent a rotation by an angle θ about the unit vector $\bar{k}$ through the origin. The rotation is then described with the unit quaternion

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\bar{k}. \tag{3}$$

which corresponds to the rotation matrix given by the Euler-Rodrigues formula $R = I + \sin\theta \bar{k}^\times + (1 - \cos\theta)\bar{k}^\times \bar{k}^\times$. This unit quaternion is also given by the exponential function

$$\exp\left(\frac{\theta\bar{k}}{2}\right) = 1 + \left(\frac{\theta\bar{k}}{2}\right) + \frac{1}{2}\left(\frac{\theta\bar{k}}{2}\right)^2 + \dots, \tag{4}$$

where $(\cdot)^i$ denotes the quaternion product of order i. The corresponding rotation matrix is given in terms of the quaternion as

$$R(q) = I + 2q_0\bar{q}^\times + 2\bar{q}^\times\bar{q}^\times. \tag{5}$$

It is noted that for $-\pi < \theta < \pi$, the scalar part of a unit quaternion can be found from

$$q_0 = \sqrt{1 - \|\bar{q}\|^2}. \tag{6}$$

B. Dual quaternions

A dual quaternion is written

$$\mathbf{q} = q_r + \varepsilon q_d, \tag{7}$$

where the real part q_r and the dual part q_d are quaternions, and ε is the dual unit, which is defined by $\varepsilon^2 = 0$ and $\varepsilon \neq 0$ [22]. Addition of two dual quaternions $\mathbf{a} = a_r + \varepsilon a_d$ and $\mathbf{b} = b_r + \varepsilon b_d$ is given by

$$\mathbf{a} + \mathbf{b} = (a_r + b_r) + \varepsilon(a_d + b_d). \tag{8}$$

The quaternion product of two dual quaternions is given by

$$\mathbf{a} \otimes \mathbf{b} = a_r \otimes b_r + \varepsilon(a_r \otimes b_d + a_d \otimes b_r). \tag{9}$$

The conjugate of a dual quaternion is $\mathbf{q}^* = q_r^* + \varepsilon q_d^*$ and the dual magnitude $\|\mathbf{q}\|$ of a dual quaternion is given by

$$\|\mathbf{q}\|^2 = \mathbf{q} \otimes \mathbf{q}^* = q_r \otimes q_r^* + \varepsilon(q_r \otimes q_d^* + q_d \otimes q_r^*). \tag{10}$$

The inverse of a dual quaternion is

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2}. \tag{11}$$

The reduced dual quaternion $\bar{\mathbf{q}} = \bar{q}_r + \varepsilon \bar{q}_d$ can be represented by the vector $[\bar{\mathbf{q}}] = [\bar{q}_r^\top\ \bar{q}_d^\top]^\top$, with the matrix form defined

$$\bar{\mathbf{q}}^\times = \begin{bmatrix} \bar{q}_r^\times & 0_{3\times3} \\ \bar{q}_d^\times & \bar{q}_r^\times \end{bmatrix}. \tag{12}$$

The unit dual quaternion is subject to the constraint $\mathbf{q} \otimes \mathbf{q}^* = 1$. The pose or displacement of a rigid body can be described as a rotation θ about a line $\bar{\mathbf{k}} = \bar{k}_r + \varepsilon \bar{k}_d$ given in Plücker coordinates, and a translation d along the line. Here $\bar{k}_r$ is the direction vector of the line, and $\bar{k}_d = \bar{p} \times \bar{k}_r$ is the moment of the line where $\bar{p}$ is the position of a point on the line [15]. The dual angle of the displacement is $\boldsymbol{\theta} = \theta + \varepsilon d$, where d is a scalar describing the translation along $\bar{\mathbf{k}}$. The unit dual quaternion describing the displacement is

$$\mathbf{q} = \cos\left(\frac{\boldsymbol{\theta}}{2}\right) + \sin\left(\frac{\boldsymbol{\theta}}{2}\right)\bar{\mathbf{k}}. \tag{13}$$

This unit dual quaternion is also given by the exponential function

$$\exp\left(\frac{\boldsymbol{\theta}\bar{\mathbf{k}}}{2}\right) = 1 + \left(\frac{\boldsymbol{\theta}\bar{\mathbf{k}}}{2}\right) + \frac{1}{2}\left(\frac{\boldsymbol{\theta}\bar{\mathbf{k}}}{2}\right)^2 + \dots, \tag{14}$$

where $(\cdot)^i$ denotes the quaternion product of order i.

The scalar parts of the reduced unit quaternion $\bar{\mathbf{q}}$ can be recovered for $-\pi < \theta < \pi$ [9] with

$$q_{r,0} = \sqrt{1 - \|\bar{q}_r\|^2} \tag{15}$$

$$q_{d,0} = \frac{-\bar{q}_r^\top \bar{q}_d}{q_{r,0}}. \tag{16}$$

The homogeneous transformation matrix $T \in SE(3)$ is given in terms of the unit dual quaternion $\mathbf{q}$ as

$$T(\mathbf{q}) = \begin{bmatrix} R(q_r) & 2q_d \otimes q_r^* \\ 0^\top & 1 \end{bmatrix}, \tag{17}$$

while the unit dual quaternion $\mathbf{q}$ can be given in terms of the transformation matrix $T(R, \bar{r})$ as

$$\mathbf{q}(T) = q(R) + \varepsilon\frac{1}{2}\bar{r} \otimes q(R). \tag{18}$$

A composite displacement $T = T_1 T_2$ will correspond to the dual quaternion $\mathbf{q} = \mathbf{q}_1\mathbf{q}_2$, where $\mathbf{q}_1$ corresponds to T_1 and $\mathbf{q}_2$ corresponds to T_2.

The kinematic differential equation for a dual quaternion can be written

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \bar{\boldsymbol{\omega}}, \tag{19}$$

where $\bar{\boldsymbol{\omega}} = \bar{\omega} + \varepsilon\bar{v}$ is the twist of the motion given in terms of the angular velocity $\bar{\omega}$ and the velocity $\bar{v}$. Suppose that the initial pose is given by a dual quaternion $\mathbf{q}_1$ and that a displacement $\mathbf{q}_2$ is made with a constant twist $\bar{\boldsymbol{\omega}}$ over a time interval h. Then $\mathbf{q}_2 = \exp(h\bar{\boldsymbol{\omega}}/2)$, and the resulting pose is given by the quaternion $\mathbf{q} = \mathbf{q}_1 \otimes \exp(h\bar{\boldsymbol{\omega}}/2)$. Note that the use of the dual quaternion product in the update of the pose ensures that the result $\mathbf{q}$ is a unit dual quaternion. To use this in time integration of the kinematic differential equations, e.g., in the time propagation of a Kalman filter, it is necessary to compute the dual quaternion exponential, or some approximation of the dual quaternion exponential like the Cayley transform.

C. The Cayley transform

The mapping of a vector $\bar{u} \in \mathbb{R}^3$ to a unit quaternion can be performed with the Cayley transform [19] and is given by

$$\mathrm{cay}(\bar{u}) \triangleq (1 + \bar{u}) \otimes (1 - \bar{u})^{-1} \tag{20}$$

$$= \frac{1 - \bar{u}^2}{1 + \bar{u}^2} + \frac{2}{1 + \bar{u}^2}\bar{u}. \tag{21}$$

Moreover, from standard trigonometric identities it follows that

$$\mathrm{cay}\left(\bar{k}\tan\frac{\theta}{4}\right) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\bar{k} = \exp\left(\frac{\theta\bar{k}}{2}\right).$$

For small angles θ the approximation $\tan\theta \approx \theta$ can be used. This gives

$$\mathrm{cay}\left(\frac{1}{2}\frac{\theta\bar{k}}{2}\right) \approx \mathrm{cay}\left(\bar{k}\tan\frac{\theta}{4}\right) = \exp\left(\frac{\theta\bar{k}}{2}\right). \tag{22}$$

It is seen that the Cayley transform can be used to calculate an approximation of the quaternion exponential function. This approximation is a rotation about the same axis $\bar{k}$, while the angle of rotation is a first-order approximation. In the case of quaternions, the exponential function can be computed efficiently from

$$\exp(\bar{u}) = \cos(\|\bar{u}\|) + \mathrm{sinc}(\|\bar{u}\|)\bar{u}, \tag{23}$$

where $\mathrm{sinc}(x) = \sin(x)/x$.

The exponential function of a twist $\bar{\mathbf{u}} = \bar{u}_r + \varepsilon\bar{u}_d$ is a unit dual quaternion and can be approximated by $\mathbf{q} = \exp(\bar{u}_r) + \varepsilon\frac{1}{2}\exp(\bar{u}_r) \otimes \bar{u}_d$ [21]. This means that the rotation is performed first, and then the translation.

The Cayley transform of a twist is given by

$$\mathrm{cay}(\bar{\mathbf{u}}) = (1 + \bar{\boldsymbol{u}}) \otimes (1 - \bar{\boldsymbol{u}})^{-1}, \tag{24}$$

where it is shown in [19] that $\mathrm{cay}(\boldsymbol{\theta}\bar{\mathbf{k}}/4)$ is an approximation of the dual quaternion exponential function $\exp(\boldsymbol{\theta}\bar{\mathbf{k}}/2)$ with the same screw axis $\bar{\mathbf{k}}$. It can be verified by direct computation that

$$(1 - \bar{\boldsymbol{u}})^{-1} = (1 - \bar{u}_r)^{-1} + \varepsilon(1 - \bar{u}_r)^{-1} \otimes \bar{u}_d \otimes (1 - \bar{u}_r)^{-1}.$$

This gives

$$\mathrm{cay}(\bar{\mathbf{u}}) = \mathrm{cay}(\bar{u}_r) + 2\varepsilon(1 - \bar{u}_r)^{-1} \otimes \bar{u}_d \otimes (1 - \bar{u}_r)^{-1}.$$

D. The Dual Quaternion Multiplicative EKF

The DQ-MEKF is derived and presented in [9]. This filter simultaneously estimates attitude and position using dual quaternions with a multiplicative measurement update. This ensures the unity constraint of the dual quaternion representing the pose of the body. The filter is of discrete-continuous type, meaning that the process is continuous while the measurements arrive at discrete time-instances.

In this paper we will only repeat the main equations of the filter and refer to [9] for a complete derivation and details. We have also chosen to follow the notation of the original authors for ease of reference.

1) Time Propagation: Consider a body frame B and a reference frame I. The pose of B with respect to I is described by the unit dual quaternion $\mathbf{q}_{B/I}$. The linear velocity $\bar{v}^B_{B/I}$ and angular velocity $\bar{\omega}^B_{B/I}$ of B described in the coordinate frame of B is represented by the dual vector

$$\bar{\boldsymbol{\omega}}^B_{B/I} = \bar{\omega}^B_{B/I} + \varepsilon\bar{v}^B_{B/I}. \tag{25}$$

Given an initial pose $\hat{\mathbf{q}}_{B/I}(t_0)$, the estimate of the pose is propagated using

$$\frac{d}{dt}\left(\hat{\mathbf{q}}_{B/I}\right) = \frac{1}{2}\hat{\mathbf{q}}_{B/I} \otimes \hat{\bar{\boldsymbol{\omega}}}^B_{B/I}, \tag{26}$$

where

$$\hat{\bar{\boldsymbol{\omega}}}^B_{B/I} = \bar{\boldsymbol{\omega}}^B_{B/I,m} - \hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}. \tag{27}$$

Here, $\bar{\boldsymbol{\omega}}^B_{B/I,m}$ is a measurement of $\bar{\boldsymbol{\omega}}^B_{B/I}$ and $\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}$ is an estimate of the dual bias vector $\bar{\mathbf{b}}_{\boldsymbol{\omega}} = \bar{b}_\omega + \varepsilon\bar{b}_v$, which represents the bias of the linear and angular velocity measurements. Given an initial bias $\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}(t_0)$, the estimate of the bias is propagated using

$$\frac{d}{dt}\left(\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}\right) = 0. \tag{28}$$

Given an initial covariance matrix $P(t_0)$ for the pose and the bias, the covariance matrix is propagated according to the Riccati equation

$$\frac{d}{dt}(P) = FP + PF^\top + GQG^\top, \tag{29}$$

where

$$F_{12\times12} = \begin{bmatrix} -\hat{\boldsymbol{\omega}}_{B/I}^{B}{}^{\times} & -\frac{1}{2}I_{6\times6} \\ 0_{6\times6} & 0_{6\times6} \end{bmatrix} \tag{30}$$

is the linearized system matrix,

$$G_{12\times12} = \begin{bmatrix} -\frac{1}{2}I_{6\times6} & 0_{6\times6} \\ 0_{6\times6} & I_{6\times6} \end{bmatrix} \tag{31}$$

is the process disturbance matrix and $Q_{12\times12}$ is the process disturbance covariance matrix. This means that the state covariance matrix P is of dimension 12×12.

2) Measurement Update: Given a measurement $\mathbf{q}_{B/I,m}$ of $\mathbf{q}_{B/I}$ at time instance t_k, the measurement sensitivity matrix is

$$H_{6\times12}(t_k) = \begin{bmatrix} I_{6\times6} & 0_{6\times6} \end{bmatrix}. \tag{32}$$

The Kalman gain $K_{12\times6}$ is then calculated from

$$K = P^- H^\top \left(HP^-H^\top + R\right)^{-1}, \tag{33}$$

where P^- denotes the predicted state covariance matrix and $R_{6\times6}$ is the measurement noise covariance matrix. The Kalman state update is then calculated as

$$\begin{bmatrix} \Delta^\star\overline{\delta\hat{\mathbf{q}}_{B/I}}(t_k) \\ \Delta^\star\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}(t_k) \end{bmatrix} = K\left[\overline{\hat{\mathbf{q}}_{B/I}^{-}(t_k)^* \otimes \mathbf{q}_{B/I,m}(t_k)}\right], \tag{34}$$

where $\hat{\mathbf{q}}_{B/I}^{-}$ denotes the predicted pose. The updated state estimate is calculated as

$$\hat{\mathbf{q}}_{B/I}^{+}(t_k) = \hat{\mathbf{q}}_{B/I}^{-}(t_k) \otimes \Delta^\star\delta\hat{\mathbf{q}}_{B/I}(t_k) \tag{35}$$

$$\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}^{+}(t_k) = \hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}^{-}(t_k) + \Delta^\star\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}(t_k), \tag{36}$$

where $\hat{\bar{\mathbf{b}}}_{\boldsymbol{\omega}}^{-}$ denotes the predicted bias. The vector part of $\Delta^\star\delta\hat{\mathbf{q}}_{B/I}(t_k)$ is given by $\Delta^\star\overline{\delta\hat{\mathbf{q}}_{B/I}}(t_k)$ and the scalar part is found from (15) and (16). The multiplicative update of $\hat{\mathbf{q}}_{B/I}$ ensures the unity constraint.

Finally the covariance matrix is updated according to

$$P^+ = (I - KH)P^-(I-KH)^\top + KRK^\top. \tag{37}$$

E. Discrete Time Propagation

In a computer implementation of the DQ-MEKF, the time propagation step has to be implemented in a discrete manner. This is not shown in [9], but it is mentioned that a normalization and projection of $\hat{\mathbf{q}}_{B/I}$ is required after each propagation step, due to the violation of the algebraic constraints of the unit dual quaternion. These violations may occur through the use of numerical integration methods such as the Euler method.

A first order application of the Euler method on (26) gives the integration scheme

$$\hat{\mathbf{q}}_{B/I}^{-}(t_k) = \hat{\mathbf{q}}_{B/I}^{+}(t_{k-1}) + \frac{h}{2}\hat{\mathbf{q}}_{B/I}^{+}(t_{k-1}) \otimes \hat{\boldsymbol{\omega}}_{B/I}^{B}(t_{k-1}), \tag{38}$$

where h is the time step. This approach violates the unit constraint of $\hat{\mathbf{q}}_{B/I}^{-}(t_k)$ due to the addition of dual quaternions.

In [6] they maintain the unit constraint of $\hat{\mathbf{q}}_{B/I}^{-}(t_k)$ using the exponential function for dual vectors in the discretization of (26). It is however not shown how this exponential function is implemented.

In this work we utilize the exponential update approximated by the Cayley transform in (24). We can then propagate (26) in a discrete manner with the integration scheme

$$\hat{\mathbf{q}}_{B/I}^{-}(t_k) = \hat{\mathbf{q}}_{B/I}^{+}(t_{k-1}) \otimes \operatorname{cay}\left(\frac{h}{4}\hat{\boldsymbol{\omega}}_{B/I}^{B}(t_{k-1})\right). \tag{39}$$

After this propagation $\hat{\mathbf{q}}_{B/I}^{-}(t_k)$ will still be a unit dual quaternion, thus normalization and projection is not required.

III. Iterative Closest Point

A set of points in Euclidean space is called a point cloud and is defined as

$$P = \{\bar{p}_i\}, \quad \bar{p}_i \in \mathbb{R}^3. \tag{40}$$

The ICP algorithm, first introduced in [2], is a method for aligning two point clouds, P_1 and P_2, that are initially placed close to each other. The point clouds should be equal, meaning that there exists a common transformation $T(R,\bar{r})$ that will exactly match all the points in one point cloud to the other

$$\bar{p}_{1,i} = R\bar{p}_{2,j} + \bar{r} \quad \forall \quad i = j. \tag{41}$$

In the case of two consecutive point clouds captured by a 3D camera, the transformation obtained by ICP can either describe the movement of the camera or the movement of a rigid object observed by the camera. In the latter case, the camera is static while the object is moving.

There are two main steps of the ICP algorithm that are repeated until convergence:

1) Find point-to-point nearest neighbour (NN) correspondences between the two point sets.
2) Minimize a error metric between the correspondences by applying a transformation.

Different approaches have been suggested to solve these steps and an overview and comparison of recognized methods can be found in [18]. One of their most significant conclusions is that the point-to-plane error metric [4] converges faster than the original point-to-point error metric. However, the use of this error metric requires the computation of surface normals, which may be computationally costly. Thus, unless surface normals are easily available, there may not be any advantage in terms of computational speed.

In a more recent review of registration algorithms for mobile robotics [17], the usefulness of ICP is highlighted by a graph showing 1800 publications on variations of the original algorithm in different experimental scenarios. This also highlights the difficulty to find a single versatile version and a common approach is to implement an adapted version suitable for the application.

Our ICP implementation is required to align noisy point clouds at high rates, as they are received from the 3D camera. We employ the constraints on the NN correspondences suggested in [20] to handle noise and different viewpoints in the point clouds. Furthermore, we choose to use the point-to-plane error metric for fast convergence. Lastly, we use the full resolution of the point clouds, as downsampling may results in loss of information and consequently loss of accuracy in the alignment.

Real-time requirements are not in the scope of this paper, however this may be a demand for some applications. It is also practical to have a fast algorithm for simulations, experiments and further work. We have therefore put emphasis on the implementation of the NN correspondences search, even though this will only affect the speed and not the accuracy of the algorithm. The NN search has a computational complexity of $\mathcal{O}(n^2)$ and is the most computationally costly operation of ICP. In addition, the internal NNs in a point cloud are needed to compute surface normals and mesh boundaries. Furthermore, these task are implemented in a parallel manner on the Graphical Processing Unit (GPU), to reduce the time of computation.

A. NN Search Between Two Consecutive Point Clouds

Given two sets of points $\{\bar{p}_{1,i}\} \in P_1$ and $\{\bar{p}_{2,j}\} \in P_2$, the indices of the NN correspondences are found according to

$$d\left(\bar{p}_{1,i}, P_2\right) = \underset{\bar{p}_{2,j} \in P_2}{\arg\min} \|\bar{p}_{2,j} - \bar{p}_{1,i}\|^2. \tag{42}$$

The set of correspondences is written as

$$C = \{(i,j) \mid \bar{p}_{1,i} \in P_1,\ \bar{p}_{2,j} \in P_2\}. \tag{43}$$

In order to reduce the complexity of the NN search and speed up the computation we exploit that a point cloud from a 3D camera is organized in a depth image matrix $\mathcal{I} \in \mathbb{R}^{N \times M}$, where each element of the matrix represent a point. The value of the z-coordinate of a point is the value of the element, while the x and y values of the point is determined by the position of the element in the matrix.

By considering a 3D camera capable of capturing point clouds at sufficiently high rates, two consecutive point clouds will overlap. This assumption, in combination with the fact that the point clouds are organized, allows for a simplified NN search. For a point $\bar{p}_{1,i} \in P1$, instead of searching through all the points in P_2, one can define a radius of pixels in the depth image $\mathcal{I}_2$ of P_2. The points represented by the pixels that fall inside the circle defined by the radius will be considered for the NN search, the rest will be ignored. The position of $\bar{p}_{1,i}$ in the depth image matrix $\mathcal{I}_1$ of P_1 is taken as the center of the circle in $\mathcal{I}_2$. The concept is illustrated in Fig. 1. This approach will greatly reduce the computational cost of the NN search, facilitating for real-time performance of ICP.

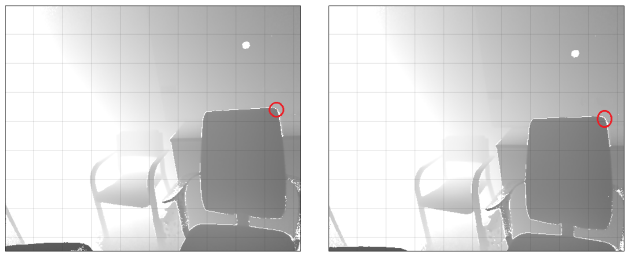
Fig. 1. Two consecutive depth images obtained at 4Hz during a hand-held movement with a Kinect v2 3D camera. The corner of the chair (marked with a circle) is observed to move 16 pixel between the two images. A search radius of 16 pixels means searching through 804 points instead of the full resolution of 217088 points.

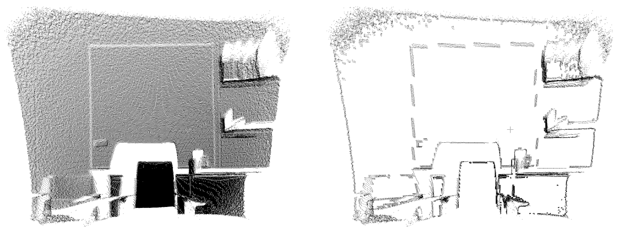
Fig. 2. The left image shows a point cloud of an office wall captured by a Kinect v2. The right image shows the detected mesh boundaries, which were obtained with the parameters $m = n = 4$ and $\sqrt{\mu_b} = 0.012$.

B. Classification of Mesh Boundaries

Fig. 2 illustrates the mesh boundaries in a point cloud captured by a Kinect v2 3D camera. The depth image matrix $\mathcal{I} \in \mathbb{R}^{Nn \times Mm}$ of a point cloud P can be subdivided into smaller sub-matrices $\mathcal{S}_i \in \mathbb{R}^{n \times m}$. By comparing the values of the elements in those sub-matrices, it can be determined whether the points represented by the elements in a specific sub-matrix are on a mesh boundary. A simple measure is the variance in z-distance, where large variance indicates that the points are on a mesh boundary.

Given a depth image matrix $\mathcal{I}_{Nn \times Mm}$, the matrix is subdivided into NM matrices with dimensions $n \times m$

$$\mathcal{S}_{1,n\times m}, \ldots, \mathcal{S}_{NM,n\times m} \subset \mathcal{I}_{Nn\times Mm}. \tag{44}$$

The points represented by $\mathcal{S}_i$ are identified as points on a mesh boundary if

$$\sigma_i^2 \geq \mu_b, \tag{45}$$

Where σ_i is the standard deviation for $\mathcal{S}_i$ and μ_b is a threshold.

C. ICP Implementation

Given two sets of points $\bar{p}_{1,i} \in P_1$ and $\bar{p}_{2,j} \in P_2$, the set of corresponding points C is found as the nearest neighbours according to (42). A point correspondence is removed from the set C if

$$d\left(\bar{p}_{1,i}, \bar{p}_{2,j}\right) \geq \mu_d, \tag{46}$$

or if either point correspondence is on a mesh boundary. Here μ_d is a distance threshold.

By using the point-to-plane error metric suggested in [4], the optimal transformation matrix $T_{\text{opt}}(R,\bar{r}) \in SE(3)$ is found by the minimization

$$T_{\text{opt}}(R,\bar{r}) = \arg\min_{R,\bar{r}} \sum_{(i,j)\in C} \left((R\bar{p}_{1,i} + \bar{r} - \bar{p}_{2,j}) \cdot \bar{n}_j \right), \tag{47}$$

where R is the rotation matrix, $\bar{r}$ is the translation and $\bar{n}_j$ is the unit surface normal of $\bar{p}_{2,j}$. The expression $\left(\tilde{R}\bar{p}_{1,i} + \bar{r} - \bar{p}_{2,j}\right)\cdot\bar{n}_j$ can be rearranged into a linear system of the form $Ax = b$, when small rotations are assumed [14]. This is solved in the least square formulation $A^\top Ax = A^\top b$.

IV. Integration of ICP and DQ-MEKF

The DQ-MEKF requires absolute pose measurements, while the ICP algorithm will provide relative pose between two point clouds. One approach is to keep track of the absolute pose by multiplying all available ICP registration results in the sequence they arrive. However, since all previous pose measurements are accounted for in a (near) optimal way in the DQ-MEKF estimate, we can multiply the current estimate with the ICP registration result to obtain an absolute measurement of the pose.

The point cloud P_{k-1} obtained at the time t_{k-1} is aligned with the point cloud P_k obtained at the time t_k using the ICP algorithm. The ICP registration result is received in the form of a transformation matrix $T(\mathbf{q}_{icp})$ which is converted to the unit dual quaternion $\mathbf{q}_{icp}$ with (18). The current pose estimate $\hat{\mathbf{q}}_{B/I}(t_{k-1})$ is then updated with $\mathbf{q}_{icp}$ to obtain an absolute pose measurement through

$$\mathbf{q}_{B/I,m}(t_k) = \hat{\mathbf{q}}_{B/I}(t_{k-1}) \otimes \mathbf{q}_{icp}. \tag{48}$$

Here the point cloud P_{k-1} is not initially aligned with a prior guess of the pose of P_k, thus we refer to this approach as ICP without initialization.

It is well known that ICP may converge to a local minimum if the initial alignment differs sufficiently from the actual. ICP also tends to converge faster if the initial alignment is accurate. As an alternative to (48), we can provide an initial alignment with the time propagated pose estimate of the DQ-MEKF filter.

We now consider the point clouds P_{k-1} and P_k, obtained at time instances t_{k-1} and t_k, with the current pose estimate $\hat{\mathbf{q}}_{B/I}(t_{k-1})$ and the propagated pose estimate $\hat{\mathbf{q}}^-_{B/I}(t_k)$. The initial pose of the ICP registration is found as

$$\mathbf{q}_{init} = \hat{\mathbf{q}}^*_{B/I}(t_{k-1}) \otimes \hat{\mathbf{q}}^-_{B/I}(t_k). \tag{49}$$

The points in P_{k-1} are transformed with $\mathbf{q}_{init}$ before registration is performed.

The ICP registration result is a correction of the predicted pose, therefore the final pose measurement is calculated as

$$\mathbf{q}_{B/I,m}(t_k) = \hat{\mathbf{q}}^-_{B/I}(t_k) \otimes \mathbf{q}_{icp}. \tag{50}$$

This approach lets us give more accurate initialization of ICP, as the camera motion is predicted by the dynamics of the filter. The architecture of our system can be seen in Fig. 3.

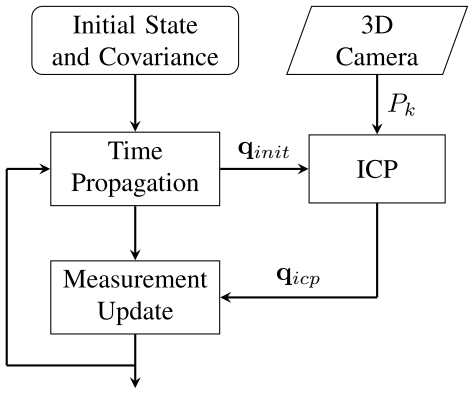

Fig. 3. Integration of ICP and DQ-MEKF where ICP is initialized with the propagated state of the filter.

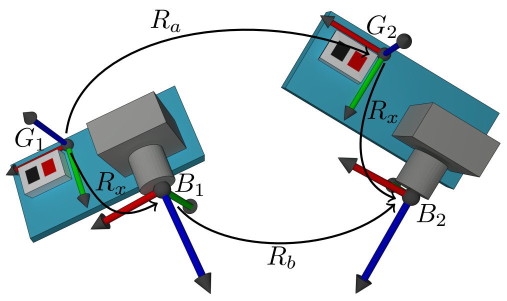

Fig. 4. The figure shows a camera-gyroscope rig at two different poses. This illustrates the kinematic chain described by (51).

V. Calibration of Gyroscope and 3D Camera

In order to include angular velocity measurements in our filter, we mounted a Bosch XDK rigidly to the camera. The Bosch XDK has a gyroscope (BMI160) capable of measuring angular velocity with an accuracy of $0.07°\,\mathrm{s}^{-1}$ at a sampling rate of 200Hz. The relative orientation between the gyroscope frame G and the camera frame B is required in order to relate the measurements in the G-frame to the B-frame. Note that only the relative orientation is needed, not the translation, as we only consider angular velocity measurements. The calibration can be performed by observing relative orientation between two time instances for each frame. Looking at Fig. 4 it can be seen that

$$R_a R_x = R_x R_b, \tag{51}$$

where $R \in SO(3)$. Here R_a is the observed rotation of the G-frame, R_b is the observed rotation of the B-frame and R_x is the relative orientation between the frames.

We obtained measurements of R_a by using the virtual orientation sensor that comes as a part of the Bosch XDK platform. Furthermore, R_b was measured by aligning consecutive point clouds from the 3D camera with ICP. Given a set of measurements $\{R_{a,i}, R_{b,i}\}$, solving (51) becomes an optimization problem on the $SO(3)$ group. We used the Lie group approach presented in [16] to find a solution for R_x.

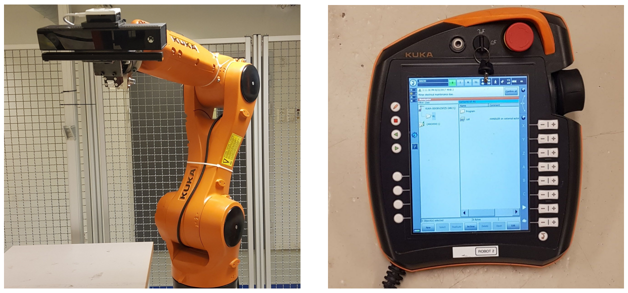
Fig. 5. Setup of the Kinect v2 camera, with the Bosch XDK, mounted on the Kuka KR6-2 robot arm to the left and the robot control pendant to the right.

Recalling that

$$\bar{\boldsymbol{\omega}}^B_{B/I,m} = \bar{\omega}^B_{B/I,m} + \varepsilon \bar{v}^B_{B/I,m}, \tag{52}$$

the measured angular velocity $\bar{\omega}^G_{B/I,m}$ obtained from the gyroscope can be included in the filter by setting

$$\bar{\omega}^B_{B/I,m} = R_x \bar{\omega}^G_{B/I,m}. \tag{53}$$

Since we do not have a measurement of $\bar{v}^B_{B/I,m}$, this is set to zero as suggested in [9].

VI. EXPERIMENTAL RESULTS

An experiment was performed to verify the proposed system. A Kinect v2 3D camera, with the Bosch XDK, was mounted on the end-effector of a Kuka KR6-2 industrial robot. The goal of the experiment was to reproduce the pose of the moving robot with our pose estimation system by obtaining point clouds from the 3D camera and angular velocity measurements from the gyroscope. The setup can be seen in Fig. 5. The ground truth for the pose was obtained by reading position and attitude of the robot from the robot control pendant at 83 Hz. The robot has a positional repeatability of ± 0.05 mm. The point clouds arrived at a rate of 5.6 Hz and the angular velocity measurements arrived at a rate of 102.4 Hz.

The filter was initialized with the states $\hat{\mathbf{q}}_{B/I}(t_0) = [1\,0\,0\,0\,0\,0\,0\,0]^\top$ and $\hat{\mathbf{b}}_{\boldsymbol{\omega}}(t_0) = [0\,0\,0\,0\,0\,0\,0\,0]^\top$, covariance $P(t_0) = 10^{-9} I_{12\times 12}$, process disturbance covariance matrix $Q = \text{diag}(0, 0, 0, 0, 0, 0, 7.5\times10^{-4}, 7.5\times10^{-4}, 7.5\times10^{-4}, 1.1\times10^{-2}, 1.1\times10^{-2}, 1.1\times10^{-2})$ (no gyroscope) and $Q_{gyro} = \text{diag}(19.6\times10^{-7}, 19.6\times10^{-7}, 19.6\times10^{-7}, 0, 0, 0, 7.5\times10^{-4}, 7.5\times10^{-4}, 7.5\times10^{-4}, 1.1\times10^{-2}, 1.1\times10^{-2}, 1.1\times10^{-2})$ (with gyroscope) and measurement noise covariance matrix $R = \text{diag}(3.513\times10^{-7}, 2.59\times10^{-6}, 3.2\times10^{-6}, 5.47\times10^{-6}, 4.98\times10^{-6}, 1.081\times10^{-4})$.

Three different configurations of the system were tested on the obtained data-set. In the first configuration the ICP algorithm was not initialized. In the second configuration the ICP algorithm was initialized with the propagated state. In the third configuration gyroscope measurements were also included in the propagation. The results for the attitude are plotted in Fig. 6 and the results for the position are plotted in Fig. 7. It can be seen that the proposed system is able to track the motion of the robot.

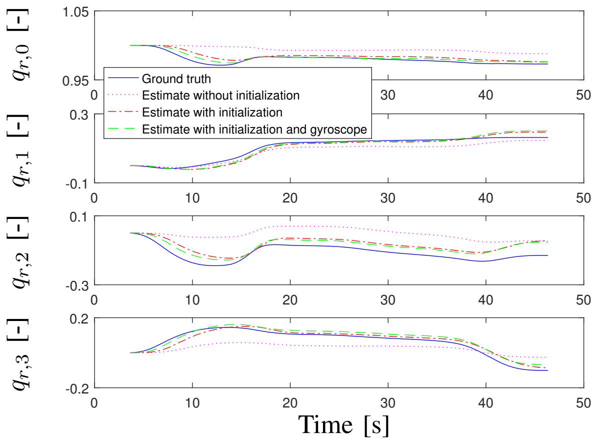

Fig. 6. True and estimated attitude. The ICP measurements were obtained with a 10 pixel NN radius search and 1 ICP iteration for each point cloud.

TABLE I

RMS DEVIATIONS FOR ESTIMATES WITH AND WITHOUT INITIALIZATION

ICP	Attitude			Position		
itera-tions	w.o. init.	w. init.	reduc-tion	w.o. init.	w. init.	reduc-tion
[#]	[°]	[°]	[%]	[mm]	[mm]	[%]
1	15.3024	6.4351	57.9	259.8965	48.7306	81.3
2	12.1043	5.8724	51.5	131.0000	33.9034	74.1
3	9.3556	5.7657	38.4	81.8587	29.9448	63.4
5	6.5884	5.7078	13.4	42.1094	28.3826	32.6
10	5.7441	5.6906	0.9	28.1942	27.7093	1.7
15	5.7081	5.6890	0.3	27.5748	27.6136	-0.1

Table I and Table II shows the Root Mean Square (RMS) deviation between the estimates and the ground truth for an increasing number of ICP iterations. ICP with (w.) and without (w.o.) initialization (init.) is compared in Table I and it can be seen that ICP with initialization performs significantly better for few ICP iterations. In Table II the filter estimates (ICP with initialization) with and without gyroscope (gyro.) measurements are compared. It can be seen that the gyroscope measurements improves the estimates, especially for few ICP iterations.

The difference of the RMS deviation for the three different configurations becomes small when the number of iterations increases. This indicate that ICP will converge to the same result, independent of the initial alignment, after an sufficient amount of iterations are performed. However, more ICP iterations require more computational power and may not be feasible in real-time applications. It can be seen from Table I and Table II that the filter with initialization and gyroscope achieves approximately the same accuracy with 1 ICP iteration, as the filter without initialization and gyroscope achieves with 10 ICP iterations.

VII. CONCLUSION

This paper presents a method for pose estimation from consecutive point clouds using the ICP algorithm in combination with the DQ-MEKF. We have showed that the pose estimate of the DQ-MEKF can remain a unit dual quaternion in the discrete time propagation of the filter by using the Cayley transform. Emphasis has been put on the implementation of the ICP algorithm such that it can provide

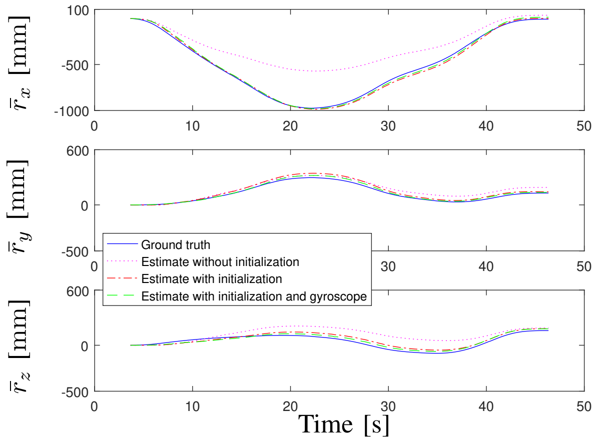

Fig. 7. True and estimated position. The position estimates are converted to the Euclidean space to give physical meaning. The ICP measurements were obtained with a 10 pixel NN radius search and 1 ICP iteration for each point cloud.

TABLE II
RMS DEVIATIONS FOR ESTIMATES WITH AND WITHOUT GYROSCOPE (ICP WITH INITIALIZATION)

ICP	Attitude			Position		
itera-tions	w.o. gyro.	w. gyro.	reduc-tion	w.o. gyro.	w. gyro.	reduc-tion
[#]	[°]	[°]	[%]	[mm]	[mm]	[%]
1	6.4351	5.7553	10.6	48.7306	31.1481	36.1
2	5.8724	5.5822	4.9	33.9034	26.5343	21.7
3	5.7657	5.5548	3.7	29.9448	25.5246	14.8
5	5.7078	5.5198	3.3	28.3826	25.0898	11.6
10	5.6906	5.5077	3.2	27.7093	24.8910	10.2
15	5.6890	5.5064	3.2	27.6136	24.8536	10.0

the filter with precise and high rate pose measurements. Furthermore, we have presented the integration of the filter with the ICP algorithm, where the initialization of the ICP algorithm is done with the difference in pose between the updated and time propagated state of the filter, which is found with the dual quaternion product.

We have performed a laboratory experiment where the estimates of our system are compared with the robot ground truth. The results show that the DQ-MEKF integrated with the ICP algorithm is able to track the motion of the robot, and that the proposed ICP initialization strategy improves the estimates. The proposed initialization reduced the RMS deviation with 57.9% in attitude and 81.3% in position for 1 ICP iteration in the experiment. The estimates were further improved by including angular velocity measurements from a gyroscope in the time propagation of the filter. We found that the gyroscope reduced the RMS deviation with 10.6% in attitude and 36.1% in position for 1 ICP iteration in the experiment. It is noted that the filter without ICP initialization and gyroscope requires 10 ICP iterations to achieve approximately the same accuracy as the filter with initialization and gyroscope achieves with 1 ICP iteration.

Finally, it is noted that the filter also provides linear and angular velocity estimates in the bias state, which might be required when implementing a control law.

ACKNOWLEDGMENT

The research presented in this paper has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

REFERENCES

[1] F. Aghili and C.-Y. Su. Robust Relative Navigation by Integration of ICP and Adaptive Kalman Filter Using Laser Scanner and IMU. *IEEE/ASME Transactions on Mechatronics*, 21(4):2015–2026, 2016.

[2] P. J. Besl and N. D. McKay. Method for registration of 3-D shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[3] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263, 2008.

[4] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[5] J. L. Crassidis, F. L. Markley, and Y. Cheng. Survey of nonlinear attitude estimation methods. *Journal of guidance, control, and dynamics*, 30(1):12–28, 2007.

[6] Y. Deng, Z. Wang, and L. Liu. Unscented kalman filter for spacecraft pose estimation using twistors. *Journal of Guidance, Control, and Dynamics*, 39(8):1844–1856, 2016.

[7] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002.

[8] O. Egeland and J. T. Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002.

[9] N. Filipe, M. Kontitsis, and P. Tsiotras. Extended Kalman filter for spacecraft pose estimation using dual quaternions. *Journal of Guidance, Control, and Dynamics*, 38(9):1625–1641, 2015.

[10] T. Hervier, S. Bonnabel, and F. Goulette. Accurate 3D maps from depth images and motion sensors via nonlinear Kalman filtering. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5291–5297. IEEE, 2012.

[11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[12] S.-G. Kim, J. L. Crassidis, Y. Cheng, A. M. Fosbury, and J. L. Junkins. Kalman filtering for relative spacecraft attitude and position estimation. *Journal of Guidance Control and Dynamics*, 30(1):133–143, 2007.

[13] E. J. Lefferts, F. L. Markley, and M. D. Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429, 1982.

[14] K.-L. Low. Linear least-squares optimization for point-to-plane ICP surface registration. *Chapel Hill, University of North Carolina*, 4, 2004.

[15] J. M. McCarthy and G. S. Soh. *Geometric design of linkages*, volume 11. Springer Science & Business Media, 2010.

[16] F. C. Park and B. J. Martin. Robot sensor calibration: solving AX= XB on the Euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994.

[17] F. Pomerleau, F. Colas, R. Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015.

[18] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[19] J. Selig. Exponential and Cayley maps for dual quaternions. *Advances in applied Clifford algebras*, 20(3):923–936, 2010.

[20] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM, 1994.

[21] X. Wang and C. Yu. Feedback linearization regulator with coupled attitude and translation dynamics based on unit dual quaternion. In *Intelligent Control (ISIC), 2010 IEEE International Symposium on*, pages 2380–2384. IEEE, 2010.

[22] Y. Wu, X. Hu, D. Hu, T. Li, and J. Lian. Strapdown inertial navigation system algorithms based on dual quaternions. *IEEE transactions on aerospace and electronic systems*, 41(1):110–132, 2005.