Robotics and Autonomous Systems 77 (2016) 66–73

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Kinematic modeling and control of a robot arm using unit dual quaternions

Erol Özgür [a,*], Youcef Mezouar [b]

[a] Université d'Auvergne, France
[b] Université Blaise Pascal, France

HIGHLIGHTS

- Kinematic modeling and pose control of multi-dof robotic arms.
- Compact and simple formulation.
- Use of unit dual quaternions and its algebra.

ARTICLE INFO

Article history:
Received 15 September 2015
Accepted 9 December 2015
Available online 17 December 2015

Keywords:
Dual quaternion
Robot arm
Kinematics
Control

ABSTRACT

This paper exploits screw theory expressed via unit dual quaternion representation and its algebra to formulate both the forward (position + velocity) kinematics and pose control of an n-dof robot arm in an efficient way. Efficiency is in less computer memory usage, in fast computation of the equations, in singularity-free representation of task space, in robustness to numerical errors, and in compactness of the representations. The formulation is simple, intuitive and straightforward to implement. We validated this formulation experimentally on a 7 dof robot arm.

1. Introduction

Unit dual quaternion (UDQ) representation of a pose (position + orientation) has been receiving a lot of attention from the robotics community both for kinematic modeling and control purposes [1–8] recently, although its storage and computational efficiencies over the homogeneous transformation matrix (HTM) have been known since more than two decades [9,10]. The study in [11] shows the superior performance of UDQ over HTM in kinematic modeling of an n-dof robot arm, and recently in [12] for the proportional control design. Other appealing advantages of UDQ are singularity free representation of the Euclidean space, robustness to numerical errors and compactness of the representation. UDQ has also been effectively used in computer graphics [13], in computer-aided design [14], in computer vision [15], in navigation [16] and so on.

The most well-known method for robot kinematics is based on the Denavit and Hartenberg (DH) notation [17] and the homogeneous transformation of the points through HTM [18]. So far all the existing works [4–6,11] on modeling of robot kinematics with UDQ continue to follow DH approach. *We think that DH wastes some capacity of UDQ, since DH's first design is based on point transformations with HTM.*

In this paper, for kinematic modeling, we followed the screw theory approach based on line transformations presented in [19], and we adapted it to unit dual quaternion representation and its algebra, since UDQ has been found as the most compact and efficient way of expressing screw displacement [9,10]. For kinematic control purposes, we used the logarithm of an error unit dual quaternion as a generalized proportional control law which is first introduced in [1] and we also analyzed its global stability in terms of the screw parameters' value ranges. Definition of the pose error between the two pose unit dual quaternions should be done through the multiplication operator of the dual quaternion algebra rather than the subtraction operator as it is done in [5,6] which is not correct (although the stability of the control law is proven). Some recent works [7,8] exploited UDQ for the design of robust control laws and for the flexible modeling of the cooperative task spaces by passing to $\Re^8$ manifold to obtain the lacking commutative property back through Hamilton operators (8×8 matrices), however leaving the computational advantages of UDQ algebra away. One might also think to use Rodrigues'

* Corresponding author.
E-mail address: erol.ozgur@udamail.fr (E. Özgür).

http://dx.doi.org/10.1016/j.robot.2015.12.005
0921-8890/

Table 1
Cost requirements for different representations of a rigid-body transformation.

Representation	Storage	Multiplications & additions	
HTM	12	64×	48+
UDQwH	8	64×	56+
TAA	7	43×	26+
UDQ	8	48×	40+

efficient rotation formula through a rigid-body pose represented by a 3D translation vector and a 4D rotation vector with Rodrigues' axis–angle parameters. We here name this representation as TAA. We remark that TAA has a singularity. Whenever the resulting angle in TAA is zero, the axis part of the rotation representation is undetermined [20]. Table 1 lists the storage and computational cost requirements for a rigid-body transformation under 4 different representations: homogeneous transformation matrices (HTM), unit dual quaternions with Hamilton operators (UDQwH), pose with Rodrigues' parameters (TAA) and unit dual quaternions (UDQ). Although TAA needs lesser storage, we note that it requires 7 trigonometric functions and 1 square-root function computations more on top of what is listed in Table 1. TAA also lacks an efficient algebra.

This paper combines efficiently all the advantages of screw theory based on UDQ and its algebra for kinematic modeling and pose control of a robot arm and experimentally validates it. Each relevant work, in the reviewed literature, somehow misses one point in combining all these together as it is discussed above. We then list the contributions of this paper as follows:

- All the advantages (*i.e.*, compactness, storage, computational efficiency, etc.) of unit dual quaternion representation and its algebra are exploited.
- The forward position kinematics (FPK), for the first time, is written in the dual space with the product of exponentials (POE) formula of screw theory by replacing the matrix exponentials with the unit dual quaternions. Everything is expressed in one single reference frame (*i.e.*, robot home frame). This makes FPK simpler and more intuitive. Consequence of this formulation is that the computation of the robot Jacobian is straightforward and fast.
- The kinematic modeling and the pose control problems of a robot arm are solved compactly with fewer number of arithmetic operations and storage requirements than many of the existing relevant approaches proposed in the robotics literature.
- Correctness of the proposed kinematic modeling and control approaches is validated experimentally on a 7 dof robot arm.
- All the variables and equations are explained clearly and without any ambiguity. That is to say, for example, a pose variable is precisely stated with in which frame it is defined and with respect to which frame it is expressed. The paper is also self-sufficient such that one can implement everything presented here without looking for any other relevant reference or book.

The rest of the paper goes on as follows: Section 2 explains the pose (position + orientation) representation of the end-effector, the forward position and velocity kinematics of the robot; Section 3 first defines the pose error, later it proposes a control law to regulate this pose error, and finally it analyzes the stability of the proposed control law; Section 4 validates experimentally the proposed kinematic modeling and control theory on the 7 dof Kuka robot arm; finally Section 5 concludes the paper.

We also note that, for better understanding of the paper, the reader can look up to Appendix for further information about the quaternions, dual numbers, and dual quaternions.

2. Kinematic modeling

2.1. Pose representation

We represent the position and orientation of the end-effector of a robot arm with a unit dual quaternion [13,15,21]:

$$\hat{\mathbf{x}} = \exp\left(\frac{\hat{\theta}}{2}\,\hat{\mathbf{s}}\right) = \cos\left(\frac{\hat{\theta}}{2}\right) + \hat{\mathbf{s}}\,\sin\left(\frac{\hat{\theta}}{2}\right) \tag{1}$$

where $\hat{\theta} \in \mathbb{D}$ and $\hat{\mathbf{s}} \in \mathbb{D}^{3\times 1}$ are respectively the dual angle and the unit dual vector of a directed 3D line:

$$\hat{\theta} = \theta + \varepsilon\, d, \qquad \hat{\mathbf{s}} = \boldsymbol{\ell} + \varepsilon\, \mathbf{m}, \quad \varepsilon^2 = 0,\ \varepsilon \neq 0. \tag{2}$$

Above, $\{\theta,\ d,\ \boldsymbol{\ell},\ \mathbf{m}\}$ are the screw displacement parameters. θ is a rotation angle around the screw axis, d is a translation along the same screw axis, $\boldsymbol{\ell}$ is the unit direction vector of this screw axis, and $\mathbf{m}$ is the moment vector of this screw axis computed with respect to the origin of the home frame of the robot arm. Eq. (1) can be rewritten in terms of a quaternion pair as follows:

$$\hat{\mathbf{x}} = \mathbf{q}_R + \varepsilon\, \mathbf{q}_T \tag{3}$$

where $\mathbf{q}_R$ is a unit quaternion for rotation and $\mathbf{q}_T$ is a quaternion for translation. These rotation and translation quaternions can be written with the known screw displacement parameters [15] as below:

$$\mathbf{q}_R \triangleq \left(\cos\left(\frac{\theta}{2}\right),\ \boldsymbol{\ell}\,\sin\left(\frac{\theta}{2}\right)\right) \tag{4}$$

$$\mathbf{q}_T \triangleq \left(-\frac{d}{2}\,\sin\left(\frac{\theta}{2}\right),\ \boldsymbol{\ell}\,\frac{d}{2}\,\cos\left(\frac{\theta}{2}\right) + \mathbf{m}\,\sin\left(\frac{\theta}{2}\right)\right). \tag{5}$$

This representation is compact, fast, stable and singularity free [9,10].

2.2. Forward position kinematics

We note here the current joint values of a robot arm with

$$\hat{\underline{\boldsymbol{\theta}}} = [\hat{\theta}_1,\ \hat{\theta}_2,\ \hat{\theta}_3,\ \ldots,\ \hat{\theta}_n]^T \in \mathbb{D}^{n\times 1}$$

and its home configuration with $\hat{\underline{\boldsymbol{\theta}}}_0 \in \mathbb{D}^{n\times 1}$. Then, for simplicity of calculations, first we move the arm to $\hat{\underline{\boldsymbol{\theta}}}_0$ and then we place the arm home frame a_0 onto the arm end-effector frame a. Thus, the relative pose between the arm home frame a_0 and the arm end-effector frame a is an identity unit dual quaternion, $\hat{\mathbf{1}}$, while $\hat{\underline{\boldsymbol{\theta}}} = \hat{\underline{\boldsymbol{\theta}}}_0$. Let $\hat{\boldsymbol{\delta}}_i$ be a unit dual quaternion which either rotates or translates (or both[1]) the end-effector frame a about the ith joint screw axis while the rest of joints are locked.

In other words, each of these unit dual quaternions $\hat{\boldsymbol{\delta}}_i$ represents the relative displacement of the end-effector frame a from the ith joint home configuration $\hat{\theta}_{i_0}$. Then, for any deviation from the home configuration, the end-effector pose of the robot arm can be calculated by multiplying all these unit dual quaternions of successive joint displacements:

$${}^{a_0}\hat{\mathbf{x}}_{a_0\,a} = {}^{a_0}\hat{\boldsymbol{\delta}}_1\ {}^{a_0}\hat{\boldsymbol{\delta}}_2\ {}^{a_0}\hat{\boldsymbol{\delta}}_3 \ldots {}^{a_0}\hat{\boldsymbol{\delta}}_n. \tag{6}$$

The resultant unit dual quaternion, ${}^{a_0}\hat{\mathbf{x}}_{a_0\,a}$, represents the new pose of the arm end-effector frame a with respect to a_0 expressed in a_0.

The order of multiplication of unit dual quaternions is important. It should be written sequentially from right to left

[1] Two joints share the same axis for rotation and translation.

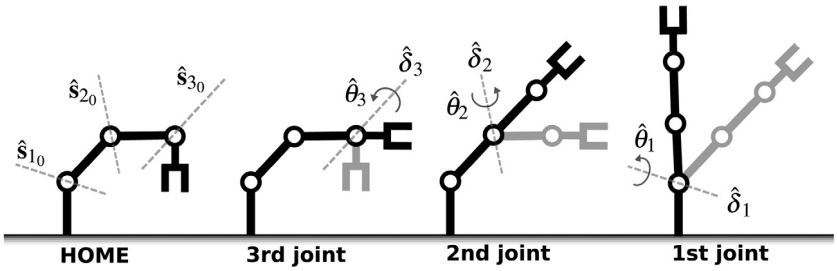

Fig. 1. A simple illustration of how forward position kinematics is applied on a 3 dof robot arm.

starting from the last joint (*i.e.*, the closest one to the end-effector, *e.g.*, here $\hat{\boldsymbol{\delta}}_n$) toward the first joint (*i.e.*, the closest one to the robot base, *e.g.*, here $\hat{\boldsymbol{\delta}}_1$).

From now on in this section, unless otherwise stated, all the variables are expressed in the robot home frame a_0.

In order to compute (6), we express a unit dual quaternion $\hat{\boldsymbol{\delta}}_i$ as follows:

$$\hat{\boldsymbol{\delta}}_i = \exp\left(\frac{\hat{\theta}_i}{2}\,\hat{\mathbf{s}}_{i_0}\right) \tag{7}$$

where the dual angle is a *relative* joint displacement with respect to the home joint position:

$$\hat{\theta}_i = \Delta\theta_i + \varepsilon\,\Delta d_i. \tag{8}$$

If a joint is revolute, then $\hat{\theta}_i = \Delta\theta_i$. If a joint is prismatic, then $\hat{\theta}_i = \varepsilon\,\Delta d_i$. The unit dual vector $\hat{\mathbf{s}}_{i_0}$ represents the joint screw axis calculated at home configuration in terms of the Plücker line coordinates:

$$\hat{\mathbf{s}}_{i_0} = \boldsymbol{\ell}_{i_0} + \varepsilon\,\mathbf{m}_{i_0} \tag{9}$$

with $\boldsymbol{\ell}_{i_0}$ the unit vector showing the direction of the joint axis, and with $\mathbf{m}_{i_0}$ the moment vector of this joint axis about the origin of the home frame:

$$\mathbf{m}_{i_0} = \mathbf{p}_{i_0} \times \boldsymbol{\ell}_{i_0}. \tag{10}$$

Here, $\mathbf{p}_{i_0}$ is a position vector from the origin of the home frame to any point lying on the joint axis (*e.g.*, calculable joint center position at home configuration). Thus, $\hat{\boldsymbol{\delta}}_i$ is a function of measurable relative joint value $\hat{\theta}_i$ and the known $\{\mathbf{p}_{i_0},\ \boldsymbol{\ell}_{i_0}\}$ at home configuration. The home configuration $\underline{\hat{\boldsymbol{\theta}}}_0$ can be chosen such that $\mathbf{p}_{i_0}$ and $\boldsymbol{\ell}_{i_0}$ are simple to write. Fig. 1 illustrates how forward position kinematics is applied gradually on a 3 dof robot arm. In Fig. 1, the left most robot's shape is chosen as home configuration, and we want to find the right most robot's end-effector pose with respect to the robot's end-effector pose at home configuration. To do so, we first compute joint displacements and then apply the unit dual quaternion transformations of these displacements successively starting from the last joint toward the first joint.

Cost analysis. An n-dof robot arm, which uses (6) to compute its forward position kinematics, needs:

$$Cost(n) = [(n-1)\,,\ \ (n-1)\,,\ n] \begin{bmatrix} 48\times \\ 40+ \\ 8f \end{bmatrix} \tag{11}$$

multiplication and addition operations and floating-point memory units. For example, a 6-dof robot arm needs $240\times$ and $200+$ operations and $48f$ memory units to compute its forward position kinematics.

If we had used Denavit–Hartenberg approach to compute the forward position kinematics of an n-dof robot arm by means of unit dual quaternions, then we would need at least $3\,n\,[48\times,\ 40+,\ 8f]^T$ more multiplication and addition operations and floating-point memory units than (11).

2.3. Forward velocity kinematics

Robot arm Jacobian relates velocities of the joint motions to the velocity of the end-effector pose:

$${}^{a_0}\hat{\underline{\boldsymbol{\xi}}}_{a_0\,a} = {}^{a_0}\hat{\mathbf{J}}\,\dot{\underline{\hat{\boldsymbol{\theta}}}} \tag{12}$$

where ${}^{a_0}\hat{\underline{\boldsymbol{\xi}}}_{a_0\,a} = \boldsymbol{\omega} + \varepsilon\,\boldsymbol{v} \in \mathbb{D}^{3\times 1}$ is a dual space velocity twist of the end-effector frame a with respect to home frame a_0 expressed in the robot home frame a_0. Above $\boldsymbol{v}$ is the translational velocity vector and $\boldsymbol{\omega}$ is the rotational velocity vector. The matrix ${}^{a_0}\hat{\mathbf{J}} \in \mathbb{D}^{3\times n}$ is the dual space Jacobian of the robot arm expressed in the home frame a_0. The dual space Jacobian ${}^{a_0}\hat{\mathbf{J}}$ is nothing else than the unit dual vectors of the joint screw axes:

$${}^{a_0}\hat{\mathbf{J}} = \begin{bmatrix} {}^{a_0}\hat{\mathbf{s}}_1 & {}^{a_0}\hat{\mathbf{s}}_2 & {}^{a_0}\hat{\mathbf{s}}_3 & \cdots & {}^{a_0}\hat{\mathbf{s}}_n \end{bmatrix} \tag{13}$$

where a unit dual vector ${}^{a_0}\hat{\mathbf{s}}_i$ expressed in the robot home frame a_0 can be computed from its known values ${}^{a_0}\hat{\mathbf{s}}_{i_0}$ at home configuration given in (9) as:

$${}^{a_0}\hat{\mathbf{s}}_i = {}^{a_0}\hat{\boldsymbol{\delta}}_{a_0\,(i-1)}\,{}^{a_0}\hat{\mathbf{s}}_{i_0}\,{}^{a_0}\hat{\boldsymbol{\delta}}^{*}_{a_0\,(i-1)} \tag{14}$$

where ${}^{a_0}\hat{\boldsymbol{\delta}}_{a_0\,(i-1)}$ represents the total displacement effect of the previous $i-1$ joints on the ith joint screw axis:

$${}^{a_0}\hat{\boldsymbol{\delta}}_{a_0\,(i-1)} = {}^{a_0}\hat{\boldsymbol{\delta}}_1\,{}^{a_0}\hat{\boldsymbol{\delta}}_2 \ldots {}^{a_0}\hat{\boldsymbol{\delta}}_{(i-1)}. \tag{15}$$

In (14) the operator $(\cdot)^*$ represents the classical quaternion conjugate of a associated dual quaternion. It is used either to transform a line [15] or to compute the inverse of a pose unit dual quaternion. Note also that in (14), if $i = 1$ then ${}^{a_0}\hat{\mathbf{s}}_1 = {}^{a_0}\hat{\mathbf{s}}_{1_0}$.

Cost analysis. An n-dof robot arm, which uses (13) to compute its Jacobian through (14), needs:

$$Cost(n) = 2\ (n-1) \begin{bmatrix} 48\times \\ 40+ \end{bmatrix} \tag{16}$$

multiplication and addition operations. For example, a 6-dof robot arm needs $480\times$ and $400+$ operations to compute its Jacobian.

Matrix–vector form representation. For the computation of the inverse velocity kinematics, one can rewrite (12) in terms of the real numbers rather than the dual numbers, and put it in matrix–vector form as below:

$${}^{a_0}\underline{\boldsymbol{\xi}}_{a_0\,a} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{M} & \mathbf{L} \end{bmatrix} \begin{bmatrix} \dot{\underline{\boldsymbol{\theta}}} \\ \dot{\underline{\mathbf{d}}} \end{bmatrix} \tag{17}$$

where $\mathbf{L} \in \Re^{3\times n}$, $\mathbf{M} \in \Re^{3\times n}$, $\underline{\boldsymbol{\theta}} \in \Re^{n\times 1}$ and $\underline{\mathbf{d}} \in \Re^{n\times 1}$ are as follows:

$$\begin{aligned} \mathbf{L} &= \begin{bmatrix} \boldsymbol{\ell}_1 & \boldsymbol{\ell}_2 & \boldsymbol{\ell}_3 & \cdots & \boldsymbol{\ell}_n \end{bmatrix}, \\ \mathbf{M} &= \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \cdots & \mathbf{m}_n \end{bmatrix} \end{aligned} \tag{18}$$

$$\begin{aligned} \underline{\boldsymbol{\theta}} &= \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \cdots & \theta_n \end{bmatrix}^T, \\ \underline{\mathbf{d}} &= \begin{bmatrix} d_1 & d_2 & d_3 & \cdots & d_n \end{bmatrix}^T. \end{aligned} \tag{19}$$

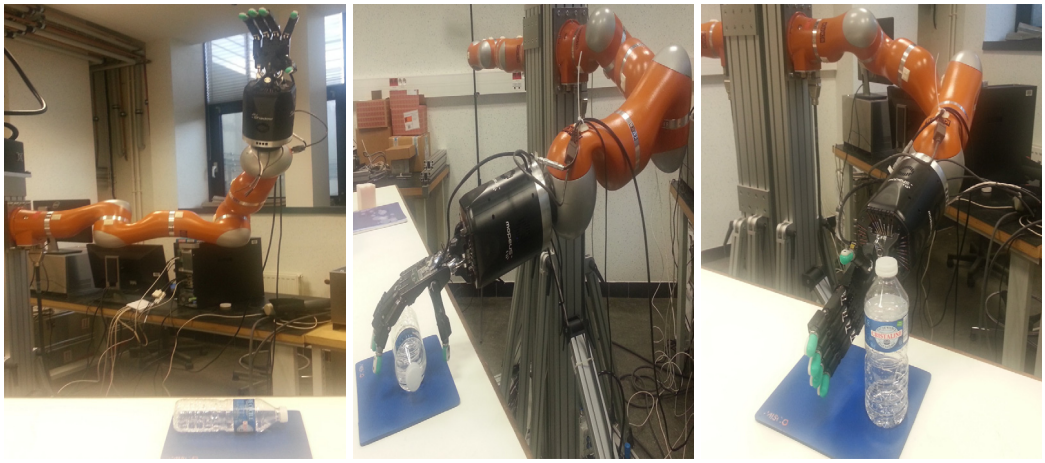

Fig. 2. Initial configuration of the robot arm and the bottle (left). Desired reach pose of the robot arm to grasp the bottle (middle). The desired corrected posture of the bottle after grasping and leaving it onto the table (right).

Notice that for a 6-dof robot arm which is composed of only revolute joints, Eq. (17) gives the well-known structure of a robot Jacobian:

$$ {}^{a_0}\underline{\boldsymbol{\xi}}_{a_0\,a} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_1 & \boldsymbol{\ell}_2 & \cdots & \boldsymbol{\ell}_6 \\ \mathbf{m}_1 & \mathbf{m}_2 & \cdots & \mathbf{m}_6 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_6 \end{bmatrix}. \tag{20} $$

One can now exploit the linear algebra algorithms on (17) to solve for the joint motions.

3. Kinematic control

3.1. End-effector pose error

We define the error unit dual quaternion, $\hat{\mathbf{e}}$, as the difference between the current end-effector pose at a and the desired end-effector pose at a_d in the home frame a_0:

$$ \hat{\mathbf{e}} = {}^{a_0}\hat{\mathbf{x}}_{a_0\,a}\,{}^{a_0}\hat{\mathbf{x}}^{*}_{a_0\,a_d} \tag{21} $$

where ${}^{a_0}\hat{\mathbf{x}}_{a_0\,a}$ is the current end-effector pose and ${}^{a_0}\hat{\mathbf{x}}^{*}_{a_0\,a_d}$ is the inverse of the desired end-effector pose ${}^{a_0}\hat{\mathbf{x}}_{a_0\,a_d}$ which is obtained through the classical quaternion conjugate of a dual quaternion.

3.2. Control law

We define the Cartesian control law ${}^{a_0}\hat{\boldsymbol{\xi}}_{a_0\,a}$ in the dual space in terms of the logarithm of the error unit dual quaternion:

$$ {}^{a_0}\hat{\boldsymbol{\xi}}_{a_0\,a} = -\lambda\, 2\ln(\hat{\mathbf{e}}) \tag{22} $$

where λ is a positive scalar control gain. The control law (22) has a global exponential convergence behavior. The proof of this behavior can be followed through the analysis of Section 3.3. Furthermore one can find a different proof in [1] for the same control law for the case of free rigid-bodies.

In the rest of this section, for the simplicity of equations, we will drop the super and subscripts of variables (*e.g.*, ${}^{a_0}\hat{\boldsymbol{\xi}}_{a_0\,a} \equiv \hat{\boldsymbol{\xi}}$).

Exploiting (1), we can rewrite (22) as:

$$ \hat{\boldsymbol{\xi}} = -\lambda\,\hat{\theta}\,\hat{\mathbf{s}} = -\lambda\,\theta\,\boldsymbol{\ell} - \varepsilon\,\lambda\,(\theta\,\mathbf{m} + d\,\boldsymbol{\ell}) \tag{23} $$

where $\{\theta,\ d,\ \boldsymbol{\ell},\ \mathbf{m}\}$ are now the screw displacement parameters obtained from the error unit dual quaternion $\hat{\mathbf{e}}$. In the next subsection, we analyze the stability of the proposed control law.

3.3. Stability analysis

In order to analyze the stability of the proposed control law, we write the following positive definite Lyapunov candidate function:

$$ V = \hat{\mathbf{e}} \circ \hat{\mathbf{e}} > 0 \tag{24} $$

where "$\circ$" is a bi-operator for vector dot product between the elements of the associated left and right dual quaternions. Afterward we differentiate this Lyapunov candidate function V with respect to time so that we can check its negative definiteness. This yields:

$$ \dot{V} = 2\,\hat{\mathbf{e}} \circ \dot{\hat{\mathbf{e}}} \tag{25} $$

where the derivative of the error unit dual quaternion, $\dot{\hat{\mathbf{e}}}$, can be rewritten in terms of the velocity twist (*i.e.*, Cartesian control law) expressed in the robot home frame (so-called spatial frame) as follows:

$$ \dot{\hat{\mathbf{e}}} = \frac{1}{2}\,\hat{\boldsymbol{\xi}}_{\wedge}\,\hat{\mathbf{e}}. \tag{26} $$

Substituting (26) into (25) yields:

$$ \dot{V} = \hat{\mathbf{e}} \circ (\hat{\boldsymbol{\xi}}_{\wedge}\,\hat{\mathbf{e}}) \tag{27} $$

where $\hat{\boldsymbol{\xi}}_{\wedge} = (0,\ \omega) + \varepsilon\,(0,\ \boldsymbol{v})$ is the Cartesian control law written in the dual quaternion space by augmenting its real and dual parts with zero scalars. Expanding (27) in terms of the screw parameters and then simplifying it, we obtain the following expression:

$$ \dot{V} = -\lambda\,\frac{1}{2}\left(d^2 + \|\mathbf{m}\|^2\,\theta\,\sin(\theta)\right). \tag{28} $$

Afterward by analyzing (28), we conclude that

$$ \text{If} \quad -\pi \le \theta \le \pi \text{ then } \dot{V} < 0. \tag{29} $$

Consequently if (29) is valid and the robot arm Jacobian (13) is non-singular, then the control law is globally exponentially stable.

4. Experiments

The presented formulation is validated on a Kuka LWR IV seven dof robot arm which is equipped with a Shadow dexterous hand [22]. In the experiment, we first reach to grasp a bottle lying on a table from a known pose, then after grasping we correct the posture of the bottle and put it back. In Fig. 2 left image shows the initial configuration the Kuka robot arm plus the Shadow

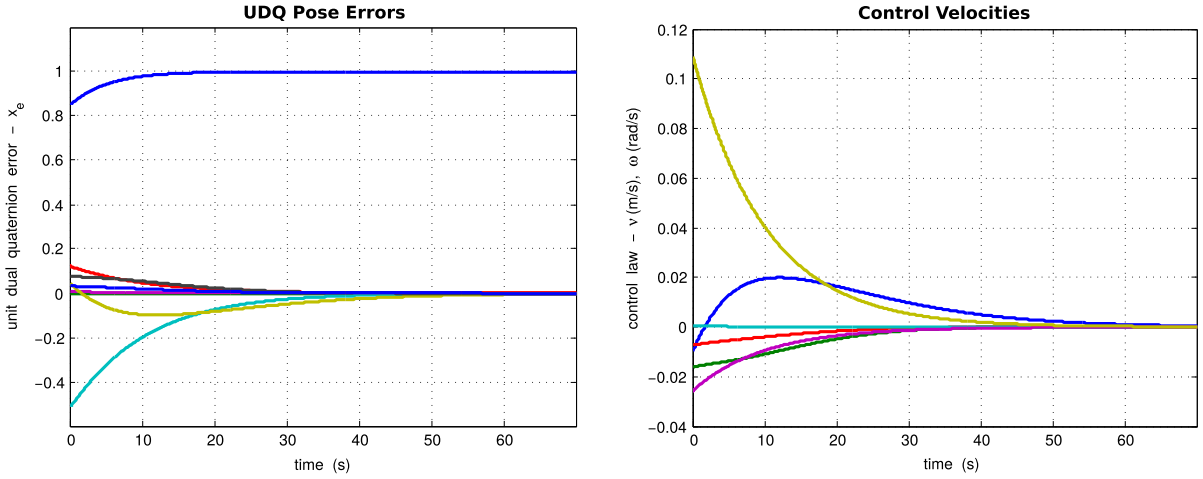

Fig. 3. Evolutions of the unit dual quaternion error (left) and the control law (right) vs. time while reaching to grasp the bottle.

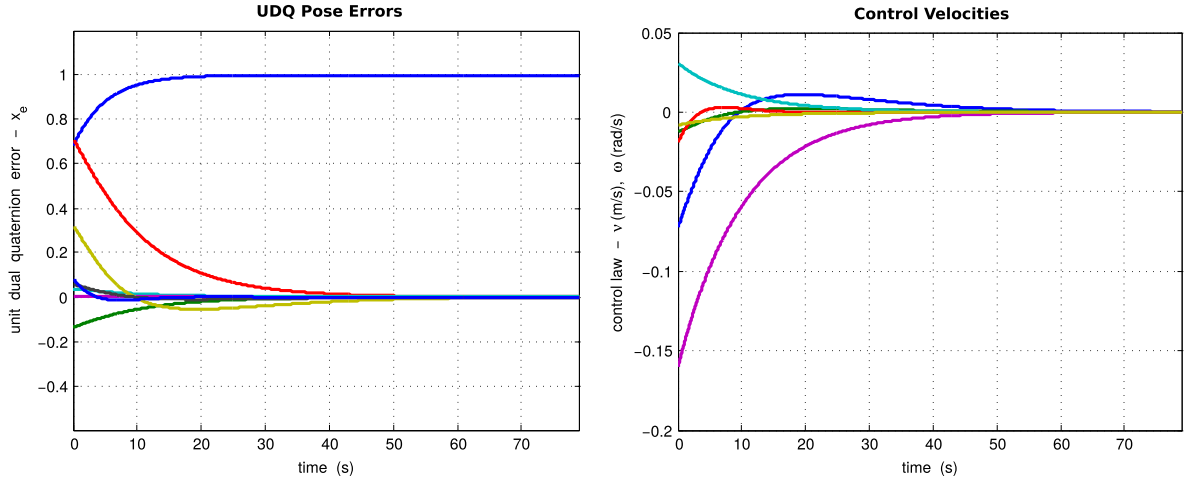

Fig. 4. Evolutions of the unit dual quaternion error (left) and the control law (right) vs. time while correcting the posture of the bottle to put it back.

dexterous hand and the bottle lying on the table. In Fig. 2 middle image shows the desired reach pose of the robot arm, and right image shows the desired corrected posture of the bottle.

Fig. 3 depicts the evolutions of the unit dual quaternion error and the control law versus time while moving to the desired reach pose shown in the middle image of Fig. 2. Fig. 4 depicts the evolutions of the unit dual quaternion error and the control law versus time while correcting the posture of the bottle toward the desired posture shown in the right image of Fig. 2. Finally, Fig. 5 shows the traces of the Cartesian poses of the end-effector registered during the whole manipulation task. One can observe from Figs. 3 and 4 that both reaching to bottle and correcting its posture tasks are successfully realized.

5. Conclusions

This paper used unit dual quaternions to model the kinematics and then to control the pose of a robot arm. Modeling is compact and fast. Therefore, computation of the control law is fast. Besides, the task space is singularity free. This formulation provides an important advantage if one uses it to model and control a robotic system which has many degrees of freedom, such as a humanoid robot.

This work may provide a basis for future research on dynamic modeling and control of robot arms in a more compact and efficient way than the existing methods using unit dual quaternions.

Appendix

A.1. Quaternions

Irish mathematician Sir William Hamilton introduced the quaternion in 1843 [23] as a geometrical operator to map two vectors to each other in 3D space. By mapping, he means reflection, rotation and scaling. Majority of applications use pure rotations. This restricts the quaternions to those with unit magnitude and that use only multiplication operation to combine different rotations. The quaternion set $\mathbb{H}$ can be seen as a four-dimensional pseudo vector space over the real numbers $\Re^4$. A quaternion $\mathbf{q} \in \mathbb{H}$ can be represented with a real scalar part $s \in \Re$ and an imaginary vector part $\mathbf{v} \in \Re^3$:

$$\mathbf{q} \triangleq (s,\ \mathbf{v}), \quad \mathbf{v} = [v_x,\ v_y,\ v_z]^T. \tag{A.1}$$

Two quaternions can be multiplied with each other as follows:

$$\mathbf{q}_1\,\mathbf{q}_2 = (s_1\,s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2,\ s_1\,\mathbf{v}_2 + s_2\,\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \tag{A.2}$$

where "$\cdot$" is the vector dot product and "$\times$" is the vector cross product. The quaternion multiplication is associative but not commutative.

Conjugate and norm. Conjugate $\mathbf{q}^*$ and norm $\|\mathbf{q}\|$ of a quaternion are given as below:

$$\mathbf{q}^* \triangleq (s,\ -\mathbf{v}) \tag{A.3}$$

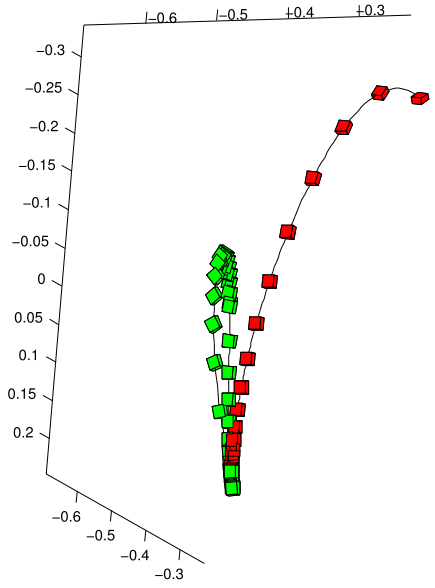

Fig. 5. Cartesian pose trajectory of the end-effector while reaching to grasp (red) and then correcting the posture (green) of the bottle to put it back. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\|\mathbf{q}\| = \sqrt{\mathbf{q}\,\mathbf{q}^*} = \sqrt{\mathbf{q}^*\,\mathbf{q}} = \sqrt{s^2 + \mathbf{v}\cdot\mathbf{v}}. \tag{A.4}$$

If $\|\mathbf{q}\| = 1$, then $\mathbf{q}$ is a unit quaternion and as well as its inverse is $\mathbf{q}^{-1} = \mathbf{q}^*$.

Rotation. One can write a 3D rotation, expressed by an angle θ around a unit vector $\boldsymbol{\ell}$, in terms of a unit quaternion as follows:

$$\mathbf{q}_R \triangleq (\cos(\theta/2),\ \sin(\theta/2)\,\boldsymbol{\ell}). \tag{A.5}$$

In order to rotate an imaginary quaternion (*i.e.*, a quaternion with zero scalar part) $\mathbf{p}_\wedge = (0,\ \mathbf{v})$ representing a vector in 3D space, one should just pre and post multiply $\mathbf{p}_\wedge$ with the unit quaternion $\mathbf{q}_R$ and its conjugate, respectively:

$$\mathbf{p}'_\wedge = \mathbf{q}_R\,\mathbf{p}_\wedge\,\mathbf{q}_R^* \tag{A.6}$$

where $\mathbf{p}'_\wedge$ is the rotated imaginary quaternion of $\mathbf{p}_\wedge$.

A.2. Dual numbers

English mathematician Sir William Clifford introduced the set of dual numbers $\mathbb{D}$ and its algebra in 1873 [24]. He defined a dual number as follows:

$$\hat{z} = a + \varepsilon\, b, \quad \varepsilon^2 = 0,\ \varepsilon \neq 0 \tag{A.7}$$

where a is the real part and b is the dual part. Geometrically a dual number can represent a 2D position vector in the dual plane. The above expression can be rewritten as follows:

$$\hat{z} = r\,(1 + \varepsilon\,\tau) \tag{A.8}$$

where the modulus $r = a$ and the argument $\tau = b/a$ for $a \neq 0$. The multiplication operation for dual numbers, once more, yields the flavor of geometric mapping:

$$\hat{z}_1\,\hat{z}_2 = r\,(1 + \varepsilon\,\tau)\,(a + \varepsilon\, b) = r\,(a + \varepsilon\,(b + a\,\tau)) \tag{A.9}$$

which scales and shears. If multiplying dual number $\hat{z}_1$ is unit (*i.e.*, $r = 1$), then the mapping is pure shearing on the 2D position vector expressed by $\hat{z}_2$. Dual numbers can also express 2D planar lines and their arbitrary motions by means of polar coordinate parameters [25].

A.3. Plücker lines as unit dual vectors

German mathematician Study defined the dual angle notation, $\hat{\theta} = \theta + \varepsilon\, d$, which relates an arbitrary 3D spatial line $\mathbf{s}$ to a given 3D spatial line $\mathbf{s}_0$ by a rotation θ about a unique axis (common normal of the two spatial lines) and with a translation d along the same axis [26]. See Fig. A.6 (left). Thus, 3-tuple of dual angles uniquely expresses a 3D spatial line with respect to axes of the reference Cartesian frame. This 3-tuple of dual angles yields a unit dual vector representation by means of Plücker coordinates [27]:

$$\hat{\mathbf{s}} = \boldsymbol{\ell} + \varepsilon\,\mathbf{m} \tag{A.10}$$

where real part $\boldsymbol{\ell}$ is the unit direction vector of the line $\mathbf{s}$, and dual part $\mathbf{m} = (\mathbf{p} \times \boldsymbol{\ell})$ is the moment of the line about the origin $\mathbf{O}$ and it is orthogonal to $\boldsymbol{\ell}$. The $\mathbf{p}$ is an arbitrary point lying on the line. See Fig. A.6 (right). The inner product of two unit dual vectors representing two skew lines (*e.g.*, $\hat{\mathbf{s}}_0$ and $\hat{\mathbf{s}}$) yields the cosine of a dual angle (*e.g.*, $\cos(\hat{\theta})$) which relates one line to the other.

A.4. Dual quaternions

A unit dual quaternion can express either the pose (both orientation and position) or the displacement of a rigid body in 3D Cartesian space. The rigid body can be displaced by multiplying its pose unit dual quaternion with the displacement unit dual quaternion. A dual quaternion is noted as a dual number with quaternion components:

$$\hat{\mathbf{x}} = \mathbf{p} + \varepsilon\,\mathbf{q} \tag{A.11}$$

where $\mathbf{p} \triangleq (s_p,\ \mathbf{v}_p)$ and $\mathbf{q} \triangleq (s_q,\ \mathbf{v}_q)$ are quaternions.

Multiplication. The multiplication of two dual quaternions yields the following equation:

$$\hat{\mathbf{x}}_1\,\hat{\mathbf{x}}_2 = \mathbf{p}_1\,\mathbf{p}_2 + \varepsilon\,(\mathbf{p}_1\,\mathbf{q}_2 + \mathbf{q}_1\,\mathbf{p}_2). \tag{A.12}$$

Conjugates. There are three different conjugates of a dual quaternion:

1. *Classical quaternion conjugate*. This is used for 3D line transformation.
$$\hat{\mathbf{x}}^* = \mathbf{p}^* + \varepsilon\,\mathbf{q}^*. \tag{A.13}$$
2. *Dual conjugate*
$$\bar{\hat{\mathbf{x}}} = \mathbf{p} - \varepsilon\,\mathbf{q}. \tag{A.14}$$
3. *Combined conjugate*. This is used for 3D point transformation.
$$\bar{\hat{\mathbf{x}}}^* = \mathbf{p}^* - \varepsilon\,\mathbf{q}^*. \tag{A.15}$$

Norm. The norm of a dual quaternion is given as:

$$\|\hat{\mathbf{x}}\| = \sqrt{\hat{\mathbf{x}}\,\hat{\mathbf{x}}^*} = \sqrt{\hat{\mathbf{x}}^*\,\hat{\mathbf{x}}} \tag{A.16}$$

$$\|\hat{\mathbf{x}}\| = \sqrt{(s_p^2 + \mathbf{v}_p\cdot\mathbf{v}_p,\ \mathbf{0}) + \varepsilon\,2\,(s_p\,s_q + \mathbf{v}_p\cdot\mathbf{v}_q,\ \mathbf{0})}. \tag{A.17}$$

If

$$s_p^2 + \mathbf{v}_p\cdot\mathbf{v}_p = 1, \qquad 2\,(s_p\,s_q + \mathbf{v}_p\cdot\mathbf{v}_q) = 0 \tag{A.18}$$

then $\|\hat{\mathbf{x}}\| = 1$. That is to say $\hat{\mathbf{x}}$ is a unit dual quaternion and its inverse is $\hat{\mathbf{x}}^{-1} = \hat{\mathbf{x}}^*$.

Displacement. One can construct a unit dual quaternion to express a displacement as follows:

$$\hat{\mathbf{x}} = \mathbf{q}_R\left(\mathbf{1} + \varepsilon\,\frac{\mathbf{t}_\wedge}{2}\right) \quad \text{or} \quad \hat{\mathbf{x}} = \left(\mathbf{1} + \varepsilon\,\frac{\mathbf{t}_\wedge}{2}\right)\mathbf{q}_R \tag{A.19}$$

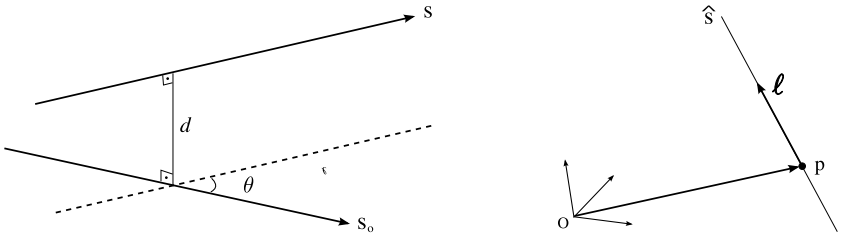

Fig. A.6. Lines—(Left): A dual angle expresses the relative pose of a line with respect to another line. (Right): Geometry of a Plücker line.

where $\mathbf{q}_R$ is a unit quaternion representing a rotation as shown in (A.5), $\mathbf{1}$ denotes an identity quaternion: $(1, \mathbf{0})$, and $\mathbf{t}_\wedge = (0, \mathbf{t})$ is the quaternion describing the translation with vector $\mathbf{t}$. Left equation in (A.19) (*resp.* right equation) first translates then rotates (*resp.* rotates then translates) a 3D geometric feature (e.g., point, line). A unit dual quaternion that only rotates ($\hat{\mathbf{x}}_R$) or that only translates ($\hat{\mathbf{x}}_T$) can then be written from (A.19) as follows:

$$\hat{\mathbf{x}}_R = \mathbf{q}_R + \varepsilon\,(0, \mathbf{0}), \qquad \hat{\mathbf{x}}_T = (1, \mathbf{0}) + \varepsilon\,\frac{\mathbf{t}_\wedge}{2} \tag{A.20}$$

and consequently the identity unit dual quaternion is $\hat{\mathbf{1}} = (1, \mathbf{0}) + \varepsilon\,(0, \mathbf{0})$. The relative displacement $\hat{\mathbf{x}}_e$ between two rigid bodies can be calculated by multiplying the pose unit dual quaternion of first rigid body with the inverse (or conjugate) of the pose unit dual quaternion of second rigid body:

$$\hat{\mathbf{x}}_e = \hat{\mathbf{x}}_1\,\hat{\mathbf{x}}_2^* \tag{A.21}$$

or the other way around.

A.5. From a finite twist to a unit dual quaternion

Let $\boldsymbol{\zeta}$ be a finite twist in $se(3)$, then it can be explicitly written with a finite rotation and a finite translation about a geometric screw line as follows [28]:

$$\begin{bmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{bmatrix} = \theta \begin{bmatrix} \mathbf{m} \\ \boldsymbol{\ell} \end{bmatrix} + d \begin{bmatrix} \boldsymbol{\ell} \\ \mathbf{0} \end{bmatrix}. \tag{A.22}$$

We can then extract the screw parameters $\{\theta, \boldsymbol{\ell}, d, \mathbf{m}\}$ of a displacement from this finite twist as below:

$$\theta = \|\boldsymbol{\omega}\|, \qquad \boldsymbol{\ell} = \frac{\boldsymbol{\omega}}{\theta}, \qquad d = \boldsymbol{\ell}^T\,\boldsymbol{\upsilon}, \qquad \mathbf{m} = \frac{1}{\theta}\,(\boldsymbol{\upsilon} - d\,\boldsymbol{\ell}). \tag{A.23}$$

Afterward, it is straightforward to write the corresponding unit dual quaternion representation, see (3) and (4).

A.6. From a unit dual quaternion to screw parameters

Let $\hat{\mathbf{x}} = \mathbf{q}_R + \varepsilon\,\mathbf{q}_T$ be a unit dual quaternion with $\mathbf{q}_R \triangleq (s_R, \mathbf{v}_R)$ and $\mathbf{q}_T \triangleq (s_T, \mathbf{v}_T)$. We can then compute the rotation angle θ as follows:

$$\theta = 2\,\arccos(s_R). \tag{A.24}$$

Afterward, we have the following two cases to compute the rest of the screw parameters:

Case when $0 < \theta < 2\,\pi$ *and* $\theta \neq 0$.

$$d = -2\,\frac{s_T}{\sin(\theta/2)} \tag{A.25}$$

$$\boldsymbol{\ell} = \frac{\mathbf{v}_R}{\sin(\theta/2)} \tag{A.26}$$

$$\mathbf{m} = \left(\mathbf{v}_T - s_R\,\frac{d}{2}\,\boldsymbol{\ell}\right)\frac{1}{\sin(\theta/2)}. \tag{A.27}$$

Case when $\theta = 0$.

$$d = 2\,\|\mathbf{v}_T\|, \qquad \boldsymbol{\ell} = 2\,\mathbf{v}_T/d, \qquad \mathbf{m} = [0,\ 0,\ 0]^T. \tag{A.28}$$

References

[1] D. Han, Q. Wei, Z. Li, Kinematic control of free rigid bodies using dual quaternions, Int. J. Autom. Comput. (2008).
[2] X. Wang, C. Yu, Unit-dual-quaternion-based PID control scheme for rigid-body transformation, in: 18th IFAC World Congress, Italy, 2011.
[3] X. Wang, D. Han, C. Yu, Z. Zheng, The geometric structure of unit dual quaternion with application in kinematic control, J. Math. Anal. Appl. (2012).
[4] M. Gouasmi, M. Ouali, F. Brahim, Robot kinematics using dual quaternions, Int. J. Robot. Autom. (2012).
[5] H. Pham, V. Perdereau, B.V. Adorno, P. Fraisse, Position and orientation control of robot manipulators using dual quaternion feedback, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.
[6] B.V. Adorno, A.P.L. Bó, P. Fraisse, P. Poignet, Towards a cooperative framework for interactive manipulation involving a human and a humanoid, in: IEEE International Conference on Robotics and Automation, 2011.
[7] L.F.C. Figueredo, B.V. Adorno, J.Y. Ishihara, G.A. Borges, Robust kinematic control of manipulator robots using dual quaternion representation, in: IEEE International Conference on Robotics and Automation, 2013.
[8] L.F.C. Figueredo, B.V. Adorno, J.Y. Ishihara, G.A. Borges, Switching strategy for flexible task execution using the cooperative dual task-space framework, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014.
[9] J. Funda, R.H. Taylor, R.P. Paul, On homogeneous transformations, quaternions, and computational efficiency, IEEE Trans. Robot. Autom. 6 (3) (1990) 382–388.
[10] J. Funda, R.P. Paul, A computational analysis of screw transformations in robotics, IEEE Trans. Robot. Autom. (1990) 348–356.
[11] N.A. Asparagethos, J.K. Dimitros, A comparative study of three methods for robot kinematics, IEEE Trans. Syst. Man Cybern. B 28 (2) (1998).
[12] X. Wang, H. Zhu, On the comparisons of unit dual quaternion and homogeneous transformation matrix, Adv. Appl. Clifford Algebr. 24 (2014) 213–229.
[13] L. Kavan, S. Collins, C. O'Sullivan, J. Zara, Dual quaternions for rigid body transformation blending, 2006.
[14] Q.J. Ge, B. Ravani, Computer aided geometric design of motion interpolants, ASME J. Mech. Des. 116 (3) (1994) 756–762.
[15] K. Daniilidis, Hand-eye calibration using dual quaternions, Int. J. Robot. Res. (1999).
[16] Y.X. Wu, X.P. Hu, D.W. Hu, J.X. Lian, Strapdown inertial navigation system algorithms based on dual quaternions, IEEE Trans. Aerosp. Electron. Syst. 41 (1) (2005) 110–132.
[17] J. Denavit, R.S. Hartenberg, A kinematic notation for the lower pair mechanism based on matrices, ASME J. Appl. Mech. (1955) 215–221.
[18] E.A. Maxwell, General Homogeneous Coordinates in Space of Three Dimensions, Cambridge University Press, Cambridge, UK, 1951.
[19] R.M. Murray, Z. Li, S.S. Sastry, A Mathematical Introduction to Robotic Manipulation, CRC Press, 1994.
[20] O.A. Bauchau, L. Trainelli, The vectorial parameterization of rotation, Nonlinear Dynam. 32 (1) (2003) 71–92.
[21] J.M. McCarthy, Introduction to Theoretical Kinematics, MIT Press, Cambridge, MA, USA, 1990.
[22] Shadow Company, Shadow Dexterous Hand C6M, Technical Specs., 2009.
[23] W.R. Hamilton, On Quaternions, or a new system of imaginaries in algebra, Phil. Mag. (1844).
[24] W.K. Clifford, Mathematical Papers, London, 1882.
[25] I.M. Yaglom, Complex Numbers in Geometry, Academic Press, 1968.
[26] E. Study, Geometrie der Dynamen, Teubner, Leipzig, 1901.
[27] J. Rooney, On the principle of transference, in: 4th World Congress on the Theory of Machines and Mechanisms, 1975.
[28] S. Stramigioli, H. Bruyninckx, Geometry and screw theory for robotics, in: Tutorial in ICRA'01, 2001.

Erol Özgür received the Ph.D. degree in Robotics and Vision from the University of Blaise Pascal, France, in 2012. Between 2012 and 2014, he was a postdoctoral fellow in Pascal Institute—UBP/CNRS/IFMA, France. Since 2015, he is an assistant professor in Université d'Auvergne. His research interests are vision-based robot control and computer vision.

Youcef Mezouar received the Ph.D. degree in Computer Science from the University of Rennes 1, France, in 2001. He was a Postdoctoral Associate in the Robotics Laboratory of the Computer Science Department, Columbia University, New York, NY. Since 2002, he has been with the Pascal Institute—UBP/CNRS/IFMA, France. His research interests are vision-based robot control and computer vision.