frontiers in
BEHAVIORAL NEUROSCIENCE

METHODS ARTICLE
published: 20 February 2013
doi: 10.3389/fnbeh.2013.00007

3D kinematics using dual quaternions: theory and applications in neuroscience

Guillaume Leclercq[1,2]*, Philippe Lefèvre[1,2] and Gunnar Blohm[3,4]*

[1] Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université Catholique de Louvain, Louvain-la-Neuve, Belgium
[2] Institute of Neuroscience, Université Catholique de Louvain, Brussels, Belgium
[3] Centre for Neuroscience Studies, Queen's University, Kingston, ON, Canada
[4] Canadian Action and Perception Network, Toronto, ON, Canada

Edited by:
Markus Lappe, Universität Münster, Germany

Reviewed by:
W. Pieter Medendorp, Radboud University Nijmegen, Netherlands
Thomas Haslwanter, Upper Austria University of Applied Sciences, Austria

***Correspondence:**
Guillaume Leclercq, Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université Catholique de Louvain, Bâtiment Euler, Avenue Georges Lemaitre, 4 1348 Louvain-la-Neuve, Belgium.
e-mail: gu.leclercq@gmail.com

Gunnar Blohm, Centre for Neuroscience Studies, Queen's University, Botterell Hall, Room 229, 18 Stuart Street, Kingston, ON, K7L 3N6, Canada.
e-mail: gunnar.blohm@queensu.ca

In behavioral neuroscience, many experiments are developed in 1 or 2 spatial dimensions, but when scientists tackle problems in 3-dimensions (3D), they often face problems or new challenges. Results obtained for lower dimensions are not always extendable in 3D. In motor planning of eye, gaze or arm movements, or sensorimotor transformation problems, the 3D kinematics of external (stimuli) or internal (body parts) must often be considered: how to describe the 3D position and orientation of these objects and link them together? We describe how dual quaternions provide a convenient way to describe the 3D kinematics for position only (point transformation) or for combined position and orientation (through line transformation), easily modeling rotations, translations or screw motions or combinations of these. We also derive expressions for the velocities of points and lines as well as the transformation velocities. Then, we apply these tools to a motor planning task for manual tracking and to the modeling of forward and inverse kinematics of a seven-dof three-link arm to show the interest of dual quaternions as a tool to build models for these kinds of applications.

Keywords: geometric algebra, forward kinematics, inverse kinematics, reference frame transformation

1. INTRODUCTION

Our environment is 3-dimensional (3D) and our body can be represented as a rigid multibody system evolving in that 3D environment, with different body parts moving relative to each other. Accurately describing the 3D kinematics of such systems is important for diverse applications in neuroscience which involve 3D kinematics: (1) forward kinematics of different 3D systems: the position and/or orientation of an end-effector (a tool held by the hand for example), (2) 3D Reference frame transformation or reference frame encoding [for example: 3D vestibulor-ocular reflex (VOR), 3D visuomotor transformation, 3D eye-head gaze shifts, 3D spatial updating, . . .], and (3) inverse kinematics: the set of joint angles corresponding to a given end-effector position and velocity. In practice, these models allow us to analyze 3D data acquired through behavioral experiments as well as to make predictions.

To model the kinematics of a rigid multibody system, we use the framework of the geometric algebra introduced by David Hestenes (see the textbook Hestenes, 1986), which easily applies to and is very convenient to model mechanical systems, in particular the kinematics. Hestenes applied this approach to some eye (Hestenes, 1994a) and arm reaching (Hestenes, 1994b) movements. Actually, he chose the geometric algebra of quaternions which have been sometimes used to model 3D eye, head and arm movements (see for example: Tweed and Vilis, 1987; Straumann et al., 1991; Hore et al., 1992; Haslwanter, 1995; Crawford and Guitton, 1997). Quaternions allow us to easily deal with rotations but not with translations. Yet we have to deal with translations since the origins of reference frames attached to each body generally do not coincide. For example, the eye rotation center is offset from the head rotation center. In order to deal with translations, in addition to the rotation, we will use the geometric algebra of dual quaternions (see Bayro-Corrochano, 2003). Dual quaternions also allow to elegantly represent a screw motion (Funda and Paul, 1990b; Aspragathos and Dimitros, 1998), defined as a combined rotation and translation along the rotation axis. They have been used in Blohm and Crawford (2007) to model combined 3D eye and head rotations in the context of 3D visuomotor transformation for reaching movements. While dual quaternions have not been widely used in the neuroscience community, they have been applied in other fields. Indeed, they have been employed with success in computer vision (Walker et al., 1991; Phong et al., 1993; Goddard and Abidi, 1998; Torsello et al., 2011) or in robotics (Daniilidis, 1999; Bayro-Corrochano et al., 2000).

The goal of this paper is to provide a tutorial of the dual quaternion geometric algebra to the neuroscientists and then to show its interests and advantages to several applications in sensorimotor control. First, we summarize the theory necessary to introduce the

different applications. The different dual quaternion operations are described and we also provide our MATLAB implementation of these operations in supplementary materials for the potential interested reader. Then we describe several applications using the dual quaternion formalism.

There are several reasons to use dual quaternions to represent the 3D kinematics. First, it is a compact and geometrically meaningful way of representing 3D rotations, translations and screw motions (Funda and Paul, 1990a; Kim and Kumar, 1990; Aspragathos and Dimitros, 1998; Daniilidis, 1999; Kavan et al., 2008). Furthermore, there are no singularities when representing 3D rotations (this is not the case if a Fick, Helmoltz or Euler representation is chosen), because a rotation is represented through a single angle θ and the axis of rotation **n** (Chou, 1992; Grassia, 1998). Moreover computational requirements (in terms of storage and speed) are lower than for other methods, including homogeneous coordinates (Funda et al., 1990; Funda and Paul, 1990b; Walker et al., 1991; Aspragathos and Dimitros, 1998), which could be important for behavioral experiments needing real-time data and processing (for example, a forward kinematic model applied to on-line raw data) but also for post-processing treatments in data analysis and model predictions. In an implementation perspective again, the neuroscience researcher concerned with the practical implementation of the transformations using dual quaternions will most likely want to consider many transformations simultaneously. For example, when analyzing experimental data, or when making predictions (from a few thousand to many more). In this context, dual quaternions are much more convenient than homogenous matrices for example. Indeed, we easily deal with an arbitrary number N of simultaneous dual quaternions transformations (using standard matrix operations), while for homogeneous coordinates, it requires the use of 3D tensors, which is possible but not easy to implement. Yet neuroscience researchers want a tool efficient and easy to use at the same time. Last but not least the dual quaternion formalism can be used to model point and lines transformations (Bayro-Corrochano, 2003), which is not the case for other formalisms like homogeneous coordinates (only point transformations). Yet applications sometimes demand that we combine both types of transformations (see section 3.3 of this article) and dual quaternions provide a unified and easy way to do that. We would have had to employ two different approaches to perform kinematic operations on points and lines if we had not used dual quaternions.

2. THEORY

In this section, we provide a tutorial or short description of the dual quaternion algebra based on the literature (Hestenes, 1994a; Bayro-Corrochano, 2003). The reader familiar with these concepts can skip this part and move to section 3. However, as far as we know of, many readers in the sensorimotor science community are not familiar with dual quaternions and therefore they will find in this section many useful dual quaternions operations which can be useful to model their 3D problem in neuroscience. Moreover, we provide a MATLAB implementation for each of these operations, for the reader who would like to use dual quaternions but who perhaps would be restrained by the implementation. This toolbox is available for download at the followings urls:

- http://www.compneurosci.com/doc/DualQuaternionToolbox.zip
- On the File exchange of Matlab Central: http://www.mathworks.com/matlabcentral/fileexchange/39288. Users are able to provide feedback, comment and rate the files.

In the following, we introduce the dual quaternion algebra, based on the quaternion algebra. Then we introduce the rotation and translation operations using the dual quaternion algebra. Afterwards we describe how points and lines can be encoded using dual quaternions, before explaining how applying the rotation or translation on these objects. Then we describe what a screw motion is and how it is encoded using a dual quaternion. Finally we briefly talk about the implementation and conversion to other formalisms.

2.1. BASICS

2.1.1. Quaternion geometric algebra

First let us define the geometric product of two 3D vectors $\vec{a}$ and $\vec{b}$ (see Hestenes, 1994a; Bayro-Corrochano, 2003). It is denoted $\vec{a}\vec{b}$ and can be expressed as:

$$\vec{a}\vec{b} \doteq \vec{a} \cdot \vec{b} + \vec{a} \wedge \vec{b} \tag{1}$$

where $\vec{a} \cdot \vec{b}$ is the classical dot product yielding a scalar number and $\vec{a} \wedge \vec{b}$ is a new entity called a bivector, resulting from the wedge product of $\vec{a}$ and $\vec{b}$. This wedge product is antisymmetric: $\vec{a} \wedge \vec{b} = -\vec{b} \wedge \vec{a}$. We will see later how to compute it in practice. Now let us consider three orthonormal basis vectors in a right-handed reference frame, $\vec{\sigma}_1$, $\vec{\sigma}_2$, and $\vec{\sigma}_3$. From the definition of the geometric product and the antisymmetric property of the wedge product, we can get that:

$$\vec{\sigma}_i\vec{\sigma}_j = \begin{cases} 1 & \text{if} \quad i = j \\ \vec{\sigma}_i \wedge \vec{\sigma}_j & \text{if} \quad i \neq j \end{cases}$$

Therefore, we obtain three basis bivectors: $\vec{\sigma}_1\vec{\sigma}_2$, $\vec{\sigma}_2\vec{\sigma}_3$, and $\vec{\sigma}_3\vec{\sigma}_1$. Then the *unit right-handed pseudoscalar* i is defined (see Hestenes, 1994a) as:

$$i = \vec{\sigma}_1\vec{\sigma}_2\vec{\sigma}_3 = \vec{\sigma}_1 \wedge \vec{\sigma}_2 \wedge \vec{\sigma}_3$$

i is thus an entity which is called a *trivector*. The i symbol is used by analogy with the complex numbers. Indeed, using the properties of the geometric product and the definition of i, we have: $i^2 = -1$. i is a duality operator since we can notice that:

$$\vec{\sigma}_1\vec{\sigma}_2 = i\vec{\sigma}_3$$
$$\vec{\sigma}_2\vec{\sigma}_3 = i\vec{\sigma}_1$$
$$\vec{\sigma}_3\vec{\sigma}_1 = i\vec{\sigma}_2$$

Therefore, the three basis bivectors are dual entities to the three basis vectors and vice-versa, which implies that for every bivector **a**, there exists a corresponding vector $\vec{a}$ such that $\mathbf{a} = i\vec{a}$. In the following, we will always use the bold notation for bivectors, and

top arrows for vectors. In particular, the bivector obtained by the wedge product can be expressed as:

$$\vec{a} \wedge \vec{b} = i(\vec{a} \times \vec{b})$$

where $\vec{a} \times \vec{b}$ is a vector representing the classical cross product of the vectors $\vec{a}$ and $\vec{b}$. A quaternion is the sum of a scalar and a bivector (Hestenes, 1994a):

$$Q = q_0 + \mathbf{q} = q_0 + i\vec{q}$$

and can be obtained as the geometric product of two vectors, as shown by Equation (1). The sum of two quaternions $A = a_0 + i\vec{a}$ and $B = b_0 + i\vec{b}$ is a quaternion $C = c_0 + i\vec{c}$ computed as follows:

$$C = A + B = \underbrace{(a_0 + b_0)}_{c_0} + i(\underbrace{\vec{a} + \vec{b}}_{\vec{c}})$$

The multiplication of two quaternions $A = a_0 + i\vec{a}$ and $B = b_0 + i\vec{b}$ is a quaternion $C = c_0 + i\vec{c}$. It is computed as (see details in **Appendix A**):

$$C = AB = (a_0 + i\vec{a})(b_0 + i\vec{b}) = \underbrace{(a_0 b_0 - \vec{a} \cdot \vec{b})}_{c_0} + i\underbrace{(a_0\vec{b} + b_0\vec{a} - \vec{a} \times \vec{b})}_{\vec{c}}$$

Other quaternion operations are also useful. The conjugate of a quaternion A is defined by:

$$A^* = (a_0 + i\vec{a})^* = a_0 - i\vec{a}$$

The norm of a quaternion A is denoted: $||A|| = \sqrt{(a_0^2 + \vec{a} \cdot \vec{a})}$. The inverse of a quaternion A is denoted A^{-1} and defined as $A^{-1} = \frac{A^*}{||A||^2}$.

2.1.2. Dual quaternion geometric algebra

A dual quaternion D is defined by

$$D \doteq D_0 + \epsilon D_1$$

where D_0 and D_1 are two quaternions and ϵ is a mathematical operator with the property that:

$$\epsilon^2 = 0 \tag{2}$$

The multiplication of two dual quaternions $D = D_0 + \epsilon D_1$ and $E = E_0 + \epsilon E_1$ is another dual quaternion $F = F_0 + \epsilon F_1$ and using the property (2), F is equal to:

$$F = DE = (D_0 + \epsilon D_1)(E_0 + \epsilon E_1) = \underbrace{D_0 E_0}_{F_0} + \epsilon(\underbrace{D_0 E_1 + D_1 E_0}_{F_1})$$

We notice that the dual quaternion multiplication involves three quaternion multiplications.

Several conjugates exist for the dual quaternion $D = D_0 + \epsilon D_1$, which are used depending on the operations needed. The first one is obtained by applying the quaternion conjugate to each quaternion composing the dual quaternion.

$$D^* = D_0^* + \epsilon D_1^*$$

The conjugate of a dual number can also be used:

$$\overline{D} = D_0 - \epsilon D_1$$

And both operations can also be combined:

$$\overline{D^*} = D_0^* - \epsilon D_1^*$$

The following identities are also useful:

$$(AB)^* = B^* A^*$$
$$(\overline{AB}) = \bar{A}\bar{B}$$
$$(\overline{AB})^* = \bar{B}^* \bar{A}^*$$

The norm of a dual quaternion D is a dual number (a dual number is of the form $a = a_0 + \epsilon a_1$ where a_0 and a_1 are real scalar numbers and ϵ is the same operator as for dual quaternions, with the property that $\epsilon^2 = 0$). The dual quaternion norm is computed in the following way (see Kavan et al., 2008):

$$||D|| = ||D_0|| + \epsilon \left[\frac{D_0^* D_1 + D_1^* D_0}{2||D_0||} \right]_{\text{scalar}}$$

where $||D_0||$ is the quaternion norm of D_0 and $[Q]_{\text{scalar}}$ is the scalar part q_0 of a quaternion $Q = q_0 + \mathbf{q}$. The inverse of a dual quaternion $A = A_0 + \epsilon A_1$ can be computed and is written (see Kavan et al., 2008):

$$A^{-1} = \frac{A^*}{||A||^2}$$

If the non-dual part, A_0, of the dual quaternion A has a zero norm, then A has no inverse.

In the next sections, the dual quaternions representing rotations, translations and or screw motions, as well as points and lines, are described using unitary dual quaternions, i.e., dual quaternions with a norm equal to 1. They are unitary under the condition that:

$$||D_0|| = 1$$
$$D_0^* D_1 + D_1^* D_0 = 0$$

Therefore, unit dual quaternions belong to a 6-dimensional manifold and are specified by six different parameters. From now on, we always use the bivector notation in order to avoid writing the i symbol at each equation. Anyway, remember that each bivector $\mathbf{a}$ is directly related to its counterpart vector $\vec{a}$ by the relation $\mathbf{a} = i\vec{a}$.

2.2. ROTATIONS AND TRANSLATIONS

In this section, we describe the dual quaternions associated with rotations and translations, as well as their velocity: rotational velocity, translational velocity. These operators will be used and combined later to carry out complex kinematic transformations.

2.2.1. Rotation

By Euler's theorem, any 3D rotation may be described by a unique unitary rotation axis **n** and a single rotation angle θ. In the dual quaternion framework, a pure rotation (whose rotation axis goes through the origin of the reference frame) is described by:

$$R_{\text{rot}} = \cos\left(\frac{\theta}{2}\right) + \mathbf{n}\sin\left(\frac{\theta}{2}\right) \tag{3}$$

where **n** is a bivector representing the unitary rotation axis coordinates expressed in the reference frame and θ is the rotation angle. Actually, R_{rot} is a quaternion, or a dual quaternion with a dual part (the part premultiplied by ϵ) equal to 0.

2.2.2. Translation

A pure translation is defined by its unitary axis **t** and its distance d. A translation dual quaternion T_{trans} is written under the following form:

$$T_{\text{trans}} = 1 + \epsilon\frac{d}{2}\mathbf{t}$$

2.2.3. Rotational velocity

Potential users should pay attention to the fact that in 3-D, the rotational velocity quaternion is not the derivative of the rotation quaternion (this is also the case if classical rotation matrices are considered). It has been shown by Hestenes (1986) that the derivative of the rotation quaternion with respect to time can be written as:

$$\dot{R}_{\text{rot}} = \frac{1}{2}R_{\text{rot}}\Omega_{\text{rot}} \tag{4}$$

where $\Omega_{\text{rot}} = \mathbf{\Omega_{rot}} = i\vec{\Omega_{\text{rot}}}$ is a rotational velocity dual quaternion which has a dual part equal to zero, and which has a non-dual part with a zero scalar component. Ω_{rot} represents the *rotational velocity*. Kinematically, the norm of $\mathbf{\Omega_{rot}}$, $||\vec{\Omega_{\text{rot}}}||$ represents the *instantaneous* angular velocity while the normalized vector $\frac{\vec{\Omega_{\text{rot}}}}{||\vec{\Omega_{\text{rot}}}||}$ represents the *instantaneous* rotation axis. Also, it can be shown that:

$$\frac{d(R^*_{\text{rot}})}{dt} = \left(\frac{dR_{\text{rot}}}{dt}\right)^* = -\frac{1}{2}\mathbf{\Omega_{rot}}R^*_{\text{rot}}$$

2.2.4. Translational velocity

The translational velocity dual quaternion is derived from the translation dual quaternion T_{trans}. Differentiating the expression of the translation dual quaternion T_{trans}, we obtain:

$$\begin{aligned}\dot{T}_{\text{trans}} &= 0 + \epsilon\left(\frac{\dot{d}(t)}{2}\mathbf{t}(t) + \frac{d(t)}{2}\dot{\mathbf{t}}(t)\right)\\ &= \epsilon\frac{\mathbf{v}}{2}\end{aligned}$$

where $\mathbf{v} \doteq \dot{d}(t)\mathbf{t}(t) + d(t)\dot{\mathbf{t}}(t)$.

2.3. DESCRIPTION OF POINT AND LINE POSITIONS AND VELOCITIES

Here we describe how we can describe either points (location and velocity) or lines (location and velocity) using the dual quaternion formalism.

2.3.1. Point position and velocity

Following what is done by Bayro-Corrochano (2003) for example, a 3D physical point P with coordinates **x** in a given reference frame R is represented by the dual quaternion $X = 1 + \epsilon\mathbf{x}$.

The velocity $\dot{\mathbf{x}}$ of point P is represented by the dual quaternion $V = \dot{X} = \epsilon\dot{\mathbf{x}}$.

2.3.2. Line position and velocity

Dual quaternions provide a convenient way to represent lines (Daniilidis, 1999; Bayro-Corrochano, 2003). A line can be represented by six coordinates (Plücker coordinates), specifying the line orientation **n** and the position of an arbitrary point of the line in space, **p**. The line representation is moreover independent of the choice of the point **p**. A line dual quaternion L is written:

$$L = \mathbf{n} + \epsilon\underbrace{\vec{p}\wedge\vec{n}}_{\doteq\mathbf{m}} \tag{5}$$

We can notice that **m** is indeed independent of the choice of **p** since the cross product $\vec{p}\times\vec{n}$ depends only on the component of $\vec{p}$ orthogonal to $\vec{n}$, and this orthogonal component is obviously the same for every point on the line L. Note also that a line dual quaternion is unitary.

The line velocity, $\dot{L}$, is obtained by computing the derivative of the line dual quaternion L:

$$\begin{aligned}\dot{L} &= \dot{\mathbf{n}} + \epsilon\dot{\mathbf{m}}\\ &= \dot{\mathbf{n}} + \epsilon\frac{d(\vec{p}\wedge\vec{n})}{dt}\\ &= \dot{\mathbf{n}} + \epsilon\left(\dot{\vec{p}}\wedge\vec{n} + \vec{p}\wedge\dot{\vec{n}}\right)\end{aligned} \tag{6}$$

We see that the line velocity dual quaternion is related to the rate of change of $\vec{n}$, as well as the rate of change of $\vec{p}$. Note that Equation (6) does not depend on the arbitrary choice of $\vec{p}$ and its rate of change (see proof in **Appendix B**).

2.4. KINEMATIC OPERATIONS ON POINTS AND LINES

Now we describe the kinematic transformations on point and lines, as the dual quaternion formalism allows us to use the same kinematic operators for point or line transformations, which is a major advantage.

Let us consider the reference frame R′, with the same origin as reference frame R, but lying in a different orientation. The transformation between both reference frames is a pure rotation with axis **n** and angle θ. Then, a point P has the following coordinates in the R′ frame (passive rotation):

$$X' = R_{\text{rot}}X\overline{R_{\text{rot}}}^* = 1 + \epsilon\left(R_{\text{rot}}\mathbf{x}\overline{R_{\text{rot}}}^*\right) \tag{7}$$

where R_{rot} is the rotation quaternion . $\overline{R_{\text{rot}}}^*$ is the conjugate quaternion: $\overline{R_{\text{rot}}}^* = \cos(\frac{\theta}{2}) - \mathbf{n}\sin(\frac{\theta}{2})$. For rotations dual quaternions, $\overline{R_{\text{rot}}}^* = R^*_{\text{rot}}$, since rotation dual quaternions have a dual component equal to zero. Therefore, in the following, we use R^*_{rot} instead of $\overline{R_{\text{rot}}}^*$ since it is shorter to write.

From another point of view, we may want to know the new position of a mobile point P after a rotation R_{rot} applied to a reference position $P_0 = 1 + \epsilon \mathbf{x}_0$ in a given reference frame R (an active rotation):

$$P = R^*_{\text{rot}} P_0 R_{\text{rot}} = 1 + \epsilon \left(R^*_{\text{rot}} \mathbf{x}_0 R_{\text{rot}} \right) \tag{8}$$

Let us now assume that reference frame R″ has the same orientation as R but R″ origin is offset from R origin. The transformation between both reference frames is a simple translation along a given unitary axis $\mathbf{t}$ and distance d. The unitary axis $\mathbf{t}$ is parallel to the line connecting the origins of frames R″ and R and is directed from R″ origin toward R origin. In R″ frame, point P has coordinates:

$$X'' = T_{\text{trans}} X \overline{T_{\text{trans}}}^* = 1 + \epsilon(\mathbf{x} + d\mathbf{t}) \tag{9}$$

where T_{trans} is the translation dual quaternion. The conjugate dual quaternion that we use for point transformation (see Bayro-Corrochano, 2003) is $\overline{T_{\text{trans}}}^* = 1 - \epsilon(-\frac{d}{2}\mathbf{t}) = T_{\text{trans}}$. We can also apply the same reasoning for the active translation of a given point.

The velocity $\dot{\mathbf{x}}$ of the 3D physical point P with coordinates $\mathbf{x}$ in a given reference frame R is represented by the dual quaternion $V \doteq \dot{X} = 0 + \epsilon\dot{\mathbf{x}}$. It can be of interest to compute the velocity of P in the frame R′, which is itself rotating compared to R. In that reference frame R′, the velocity dual quaternion V' is:

$$\begin{aligned} V' &= \frac{d(R_{\text{rot}} X R^*_{\text{rot}})}{dt} \\ &= \dot{R_{\text{rot}}} X R^*_{\text{rot}} + R_{\text{rot}} X \dot{R^*_{\text{rot}}} + R_{\text{rot}} \dot{X} R^*_{\text{rot}} \end{aligned} \tag{10}$$

Using relationship (4), expression (10) may be developed:

$$V' = R_{\text{rot}} \left(\frac{1}{2} \left(\boldsymbol{\Omega}_{\mathbf{rot}} X - X \boldsymbol{\Omega}_{\mathbf{rot}} \right) + V \right) R^*_{\text{rot}}$$

We can observe that if frame R′ does not move compared to frame R, $\boldsymbol{\Omega}_{\mathbf{rot}} = \mathbf{0}$, then the only difference is that V' has its coordinates expressed in the R′ frame instead of the R frame.

From another point of view, we may want to know the velocity $\dot{P}$ of a mobile point P during a rotation $R_{\text{rot}}(t)$ applied to a reference position $P_0 = 1 + \epsilon \mathbf{x}_0$ with reference velocity $\dot{P}_0$ in a given reference frame R (an active rotation):

$$\begin{aligned} \dot{P} &= \frac{d(R^*_{\text{rot}} P_0 R_{\text{rot}})}{dt} \\ &= \frac{1}{2} \left(P \, \boldsymbol{\Omega}_{\mathbf{rot}} - \boldsymbol{\Omega}_{\mathbf{rot}} P \right) + R^*_{\text{rot}} \dot{P}_0 R_{\text{rot}} \end{aligned}$$

Let us now compute the velocity V'' of point P in the reference frame R″ that is translated with respect to R. The transformation between both reference frames is a simple translation motion along a given unitary axis $\mathbf{t}$ and distance $d(t)$. In R″ frame, the velocity of point P is:

$$\begin{aligned} V'' &= \frac{d(T_{\text{trans}} X T_{\text{trans}})}{dt} \\ &= \underbrace{\dot{T_{\text{trans}}} X T_{\text{trans}} + T_{\text{trans}} X \dot{T_{\text{trans}}}}_{=\epsilon \dot{d}\mathbf{t}} + \underbrace{T_{\text{trans}} \dot{X} T_{\text{trans}}}_{=\dot{X}} \\ &= 0 + \epsilon(\dot{\mathbf{x}} + \dot{d}\mathbf{t}) \end{aligned} \tag{11}$$

For some applications, we may be interested in applying the kinematic operators to lines instead of points. For example, we may be interested in the orientation of an end-effector (in addition to the position) and/or the way this orientation changes with time. Lines provide a useful way to answer those questions. An example will be described in the applications. The operations are exactly identical to those described for point transformations except that another dual quaternion conjugate is used: the quaternion conjugate, A^*, instead of the mixed conjugate, $\overline{A^*}$ (see Equations 7, 9). This slight difference is due to the way that points and lines are encoded with dual quaternions (Bayro-Corrochano, 2003).

2.5. SCREW MOTION

Before moving to the applications, we describe a last (less known) kinematic operator: a screw displacement, or screw motion. First we explain what a screw motion is and how this kinematic operation is encoded with dual quaternions. Then we derive the screw motion velocity. Finally we show how screw motion can be applied easily to both points and lines.

The reader should be aware that the screw motion is not a revolutionary kinematic operator, but it is a simple alternative, providing a single dual quaternion directly for an operation which can be also obtained by combining a translation along and a rotation around an axis line which is offset from the origin. Whatever the approach chosen by the user, the final result for your application will be the same. However, screw motion dual quaternions provide a more compact way to write the kinematics for this type of movement.

2.5.1. Screw motion definition

A screw motion describes a rotation of angle θ about a line whose direction is specified by a unitary axis $\mathbf{n}$ and whose location can be described by an arbitrarily chosen point on the line, with coordinates $\mathbf{a}$, followed/combined by a translation of distance d along this axis $\mathbf{n}$. $\mathbf{a}$ is an arbitrary point of the line and this line does not necessarily go through the origin, such that the screw motion formalism also can be used to describe rotations about eccentric points—in contrast to quaternions and rotation matrices, which can only characterize rotations about the origin. A screw motion can be represented by the following dual quaternion:

$$M = \left(\cos\frac{\theta}{2} + \mathbf{n} \sin\frac{\theta}{2} \right) + \epsilon \left(-\frac{d}{2} \sin\frac{\theta}{2} + \mathbf{n}\frac{d}{2} \cos\frac{\theta}{2} + (\vec{a} \wedge \vec{n}) \sin\frac{\theta}{2} \right) \tag{12}$$

Note that the quantity $\vec{a} \wedge \vec{n}$ is independent of the vector $\mathbf{a}$ we choose on the line to describe the line location. Equation (12)

is derived (see Daniilidis, 1999; Bayro-Corrochano, 2003) by starting from the rotation quaternion R describing a rotation of angle θ around the unitary axis $\mathbf{n}$. However, as previously noted, the quaternion formulation implicitly assumes that the rotation axis line passes through the reference frame origin. If the rotation line passes through a point with coordinates $\mathbf{a}$, a general expression for the rotation dual quaternion whose axis is offset by $\mathbf{a}$, R_T, is obtained by applying a translation operator $T_L = 1 + \epsilon\frac{\mathbf{a}}{2}$ to the quaternion R, by left-multiplying it by the conjugate of T_L and right-multiply it by T_L. It is quite similar to the translation operation that we apply to lines (instead of rotation dual quaternion here) and that we described in section 2.4 :

$$\begin{aligned} R_T &= T_L^* R T_L \\ &= \left(1 - \epsilon\frac{\mathbf{a}}{2}\right)\left(\cos\frac{\theta}{2} + \mathbf{n}\sin\frac{\theta}{2}\right)\left(1 + \epsilon\frac{\mathbf{a}}{2}\right) \\ &= \left(\cos\frac{\theta}{2} + \mathbf{n}\sin\frac{\theta}{2}\right) + \epsilon\left((\vec{a}\wedge\vec{n})\sin\frac{\theta}{2}\right) \end{aligned}$$

The dual quaternion R_T describes the same rotation but takes the offset between the reference frame origin and the line into account. R_T is a transformation of R and therefore T_L had to be applied from both sides. To obtain the screw motion dual quaternion M, we need to combine a translation $T = 1 + \epsilon\frac{d}{2}\mathbf{n}$ along the line axis to the rotation R_T (here we combine operators instead of transforming one operator, so there is only a simple left-multiplication):

$$\begin{aligned} M = TR_T &= \left(\cos\frac{\theta}{2} + \mathbf{n}\sin\frac{\theta}{2}\right) \\ &+ \epsilon\left(-\frac{d}{2}\sin\frac{\theta}{2} + \mathbf{n}\frac{d}{2}\cos\frac{\theta}{2} + (\vec{a}\wedge\vec{n})\sin\frac{\theta}{2}\right) \end{aligned}$$

2.5.2. Screw motion velocity

The screw motion velocity dual quaternion, $\dot{M}$, is derived from the screw motion dual quaternion expression, M, in Equation (12). Indeed, a screw motion can be decomposed into a translational part and a rotational part, as shown in section 2.5.1. We have:

$$\begin{aligned} M &= TR_T \\ &= TT_L^* R T_L \end{aligned}$$

Therefore,

$$\begin{aligned} \dot{M} &= \dot{T}T_L^* R T_L + T\dot{T}_L^* R T_L + TT_L^* R\dot{T}_L + TT_L^*\dot{R}T_L \\ &= \underbrace{\dot{T}R}_{\text{translational term}} + \underbrace{\dot{T}_L^* R + R\dot{T}_L}_{\text{offset term}} + \underbrace{\frac{1}{2}TT_L^* R\boldsymbol{\Omega}T_L}_{\text{rotational term}} \end{aligned} \quad (13)$$

where the second line is obtained through a simple calculation (translation has no effect on a velocity dual quaternion). We can observe the contributions of several terms related to the translational velocity along the screw axis, the translational velocity of the screw axis itself and the rotational velocity term.

Similarly to rotation and translation operations, we may easily apply screw motion operations on points or lines.

2.5.3. Screw motion applied to points

Let us consider now an active screw motion $M = R_T T$ applied to a reference point P_0. The resulting position is then:

$$\begin{aligned} P &= \overline{M}^* P_0 M \\ &= T\overline{R_T}^* P_0 R_T T \end{aligned} \quad (14)$$

If we consider again the active screw motion $M = R_T T$ applied to the reference point P_0 with velocity $\dot{P}_0$, the resulting velocity is then:

$$\begin{aligned} \dot{P} &= \frac{d(\overline{M}^* P_0 M)}{dt} \\ &= \dot{\overline{M}}^* P_0 M + \overline{M}^* P_0 \dot{M} + \overline{M}^* \dot{P}_0 M \end{aligned}$$

As will be seen in the applications, we can linearly combine several rotations, translations and screw motions in a compact expression, facilitating the geometrical interpretation and the clarity.

2.5.4. Screw motion applied to lines

Let us describe the final parameters of the line $L = \mathbf{n} + \epsilon\mathbf{m}$ resulting from applying a screw motion described by $M = R_T T$ to a reference line $L_0 = \mathbf{n}_0 + \epsilon\mathbf{m}_0$. Compared to the expression used for points (see Equation 14), the only slight difference is the use of another conjugate, as previously mentioned (see Bayro-Corrochano, 2003).

$$\begin{aligned} L &= M^* L_0 M \\ &= T^* R_T^* L_0 R_T T \end{aligned}$$

Knowing the structure of a line dual quaternion (see Equation 5), we can extract the line parameters $\mathbf{n}$ and $\mathbf{m}$. Since $\mathbf{m} = \vec{p}\wedge\vec{n}$ where $\vec{p}$ is any point located on line L, we can choose a particular $\vec{p}$ by adding an additional constraint. An example would be to take the point $\vec{p}$ which is the closest to the origin.

We can also compute the line velocity $\dot{L} = \dot{\mathbf{n}} + \epsilon\dot{\mathbf{m}}$ resulting from a screw motion M to a reference line L_0 with velocity $\dot{L}_0$.

$$\begin{aligned} \dot{L} &= \frac{d(M^* L_0 M)}{dt} \\ &= \dot{M}^* L_0 M + M^* L_0 \dot{M} + M^* \dot{L}_0 M \end{aligned}$$

Again, we can retrieve the parameters $\dot{\mathbf{n}}$ and $\dot{\mathbf{m}}$. Having retrieved a particular point $\mathbf{p}$ of the line, we can extract its velocity (orthogonally to the line orientation $\mathbf{n}$).

In the **Appendix C**, we also describe some useful dual quaternions identities which may be used in the diverse sensorimotor applications that we will present in section 3. In the next section, we discuss the implementation of dual quaternions.

2.6. IMPLEMENTATION OF DUAL QUATERNIONS

From an implementation perspective, we can write a quaternion $A = a_0 + i\vec{a}$ as a four-element vector A_{tab}:

$$A_{\text{tab}} = \begin{pmatrix} a_0 \\ \vec{a} \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

where $\vec{a} = (a_1\ a_2\ a_3)^T$ where ^{T} is the *transpose* matrix operator. If we consider N quaternions simultaneously, we just create a $4 \times N$ matrix where each column is a the vector representation of the corresponding quaternion.

Using this representation, the quaternion multiplication is implemented using matricial operations:

$$A_{\text{tab}}B_{\text{tab}} = \begin{pmatrix} a_0b_0 - \vec{a}\cdot\vec{b} \\ a_0\vec{b} + b_0\vec{a} - \vec{a}\times\vec{b} \end{pmatrix} \tag{15}$$

We can also consider N simultaneous multiplications of two quaternions. This is achieved by adapting the above formula to matrices instead of vectors. These simultaneous operations are not just for theoretical purposes or generalizations. Indeed, again addressing to the neuroscientists communities, we often have to deal with hundreds or thousands of trials in behavioral experiments. Often, we compare our data to model predictions. In this framework, let us consider that we use quaternions for our model predictions. In this case, it is much faster to transform N quaternions (corresponding to N trials) simultaneously than to run them individually through a *for* loop.

A dual quaternion $D = D_0 + \epsilon D_1 = d_0 + i\vec{d}_0 + \epsilon(d_1 + i\vec{d}_1)$ can be represented using a 8-dimensional vector D_{tab} (which can also be considered as the juxtaposition of two 4-dimensional vectors representing the two quaternion components of the dual quaternion):

$$D_{\text{vec}} = \begin{pmatrix} (D_0)_{\text{tab}} \\ (D_1)_{\text{tab}} \end{pmatrix} = \begin{pmatrix} d_0 \\ \vec{d}_0 \\ d_1 \\ \vec{d}_1 \end{pmatrix}$$

Again, we can generalize to N dual quaternions by using a $8 \times N$ matrix. The dual quaternion multiplication is implemented using matricial operations and the quaternion multiplication defined in Equation (15):

$$F_{\text{vec}} = \begin{pmatrix} (D_0)_{\text{tab}}(E_0)_{\text{tab}} \\ (D_0)_{\text{tab}}(E_1)_{\text{tab}} + (D_1)_{\text{tab}}(E_0)_{\text{tab}} \end{pmatrix}$$

We will not describe how all dual quaternion operators and transformation are implemented, as most of them are easy or just a consequence of the encoding of a general dual quaternion that we just described. However, for the reader who would not like to implement herself/himself all the dual quaternion operations in her/his favorite language, we provide in supplementary materials a dual quaternion toolbox written in MATLAB providing functions implementing all the dual quaternions operations previously mentioned. Furthermore, an example and a read me file are also available. A document listing several quaternion and dual quaternion Matlab toolboxes developed by others is also provided. In this way, the potential user has access to our toolbox but also to others, and therefore he can judge which one is the most suitable for him/herself. We wanted to develop our own Matlab toolbox because the other ones did not gather all the functionalities we needed.

After introducing the dual quaternion algebra in this first part and how it can be applied in kinematics, we now move to the second part of this manuscript. This part describes how dual quaternions and their derivatives can be used to easily describe several applications from sensorimotor control in neuroscience.

3. APPLICATIONS

The dual quaternion algebra is very convenient to express the motion of rigid bodies, especially our body parts. In the following, we describe several applications of this theory. First we describe the reference frame transformations required in the 3D visuomotor transformation for reaching and tracking movements. Then we describe the forward and inverse kinematics problem for a two-link arm focusing only on the end-effector position. For that problem, we will use point transformations. Finally we focus on the forward and inverse kinematics problem for a three-link arm holding a tool whose position and orientation matters. We will use both point and line transformations. One huge advantage of dual quaternions is the fact that we can use them for both point and line transformations, which is not the case with other formalisms (homogeneous matrices are not suitable for line transformation for example, while dual matrices were developed to tackle line transformations and not point transformations).

3.1. REFERENCE FRAME TRANSFORMATION: MOVEMENT PLANNING

Here we study reference frame transformations in the context of visuomotor transformations. For instance, visually guided arm movements to reach for a seen object. Indeed, the brain has to transform the visual information about the target of interest into a set of motor commands for the arm muscles. To this end, the brain plans the movement ahead (see Shadmehr and Wise, 2005). However, the transformation between retinal information and the spatial motor plan is not trivial (see Blohm and Crawford, 2007), since our eyes and head move relative to the body (and thus relative to the shoulder, insertion point of the arm). For instance, let us consider the motion of a target in space. If the head is rolled toward the left or right shoulder, the projection of the spatial motion onto the retina will be different depending on the head roll angle (Leclercq et al., 2012). Therefore, the brain should take the head roll into account in order to generate an accurate motor plan for the arm. This transformation amounts to expressing the retinal motion into a spatial motion, thus it is a reference frame transformation problem.

In the following we describe two transformations. First we describe how we express the retinal position and motion of a target, e.g., a tennis ball, as a function of the spatial trajectory of this target and the 3D eye-head-shoulder geometry. This transformation is useful for a neuroscientist dealing with behavioral experiments. Indeed, the spatial target position and velocity are

often specified by designing the experiment (by choice of the experimenter), while the retinal position and motion can not be measured directly. These signals need to be estimated using the transformation model that we develop in this first part and measured and/or known signals about the 3D eye-head-shoulder kinematics. Then we describe the *inverse* transformation which computes the spatial position and velocity of a target from the knowledge of the retinal position and velocity as well as the 3D eye-head-shoulder geometry. This transformation is important from a neuroscience perspective as it is the transformation that the brain should implement in order to generate a spatially accurate arm movement (see Blohm and Crawford, 2007; Leclercq et al., 2012). Furthermore, using this theoretical transformation, we can easily test hypotheses about the availability, accuracy or precision of a signal (retinal and/or extra-retinal) in the brain. For that last application, we use the retinal position and velocity estimated through the first transformation described above.

In this context, dual quaternions provide a useful tool to express these transformations since they provide a geometrically meaningful way of expressing them, and they are easily implemented using a dual quaternion toolbox (as we provide it in the supplementary materials).

3.1.1. Computing the retinal position and motion

Figure 1 depicts a situation where a subject is confronted with a moving object (e.g., a ball), the target (denoted P), and she/he is going to interact with it (track or catch the target). Here we describe how the retinal position and velocity of the target are expressed as a function of the spatial position and velocity, using the dual quaternion formalism to express the 3D kinematics. In an experiment, we typically specify P position and velocity in a spatial reference frame, for example, the environment reference frame, or the shoulder reference frame in this case since we assume the shoulder remains fixed in the environment. So we need to estimate what the position and the velocity of the projection of P onto the retina are.

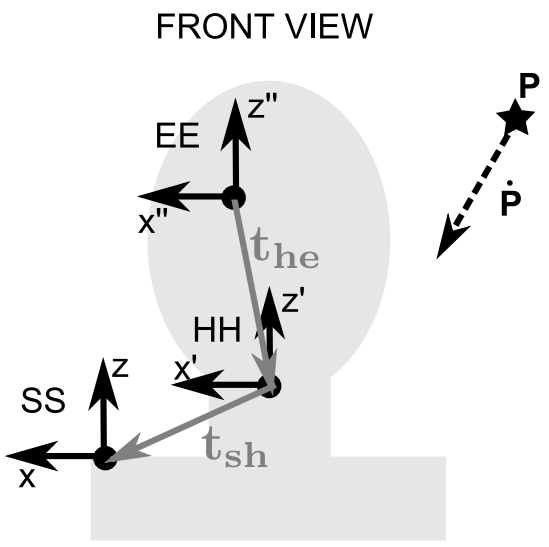

FIGURE 1 | The picture represents a front view of the eye-head-shoulder system. The pointing target position (black star) and velocity (dashed arrow) are represented. These position and velocity can be represented in different reference frames, represented on the figure. These reference frames are: SS (shoulder-centered shoulder-fixed), HH (head-centered head-fixed), and EE (eye-centered eye-fixed). The offsets between the rotation centers are also represented: t_{he} the head position in eye-centered head-fixed coordinates, and t_{sh} the shoulder position in head-centered shoulder-fixed coordinates.

In order to compute the P projection trajectory onto the retina, it is obvious that we need to know P kinematics in shoulder (space) coordinates and also what the kinematics of the eye-head-shoulder rigid body system are. In addition to a spatial reference frame, a right-handed orthonormal reference frame is attached to each rigid body (the head, the eye and the shoulder in this example, see **Figure 1**). Let us assume that P trajectory is known in the shoulder reference frame, $P^{SS}(t) = (P^{SS}{}_x, P^{SS}{}_y, P^{SS}{}_z)$. $P^{SS}(t)$ refers to P position in the **S**houlder-centered **S**houlder-fixed reference frame (*-centered* refers to the reference frame origin while *-fixed* denotes that the orientation of the axes is constant with respect to the specified rigid body) as a function of time.

The first step is to compute P trajectory (position and velocity) in an eye-centered eye-fixed reference frame, denoted $P^{EE}(t)$ and $P^{\dot{E}E}$. By combining eye-in-head rotation (R_{EH} quaternion), offset between eye and head rotation centers (T_{HE} dual quaternion), head-on-shoulder rotation (R_{HS} quaternion), and offset between head and shoulder rotation centers (T_{SH} dual quaternion), we obtain the following expression for P position in eye-centered eye-fixed coordinates as a function of P position in shoulder-centered shoulder-fixed coordinates (using point transformations, as described in section 2.4; see also **Appendix E** for the derivation).

$$P^{EE} = R_{EH} T_{HE} R_{HS} T_{SH} P^{SS} T_{SH} R^*_{HS} T_{HE} R^*_{EH} \tag{16}$$

where the rotation quaternions are expressed as a function of the rotation angles and unitary axis.

For the P velocity, we differentiate Equation (16). And after a few calculations (see **Appendix E**), we obtain:

$$\begin{aligned} P^{\dot{E}E} = & \frac{1}{2} R_{EH} (\mathbf{\Omega_{EH}} P^{EH} - P^{EH} \mathbf{\Omega_{EH}}) R^*_{EH} \\ & + \frac{1}{2} R_{EH} R_{HS} (\mathbf{\Omega_{HS}} P^{HS} - P^{HS} \mathbf{\Omega_{HS}}) R^*_{HS} R^*_{EH} \\ & + R_{EH} R_{HS} P^{\dot{S}S} R^*_{HS} R^*_{EH} \end{aligned} \tag{17}$$

where P^{EH} (resp.P^{HS}) is the target position expressed in eye-centered head-fixed (resp. head-centered shoulder-fixed) coordinates. They can be computed, similarly to the expression in Equation (16), as:

$$\begin{aligned} P^{EH} &= T_{HE} R_{HS} T_{SH} P^{SS} T_{SH} R^*_{HS} T_{HE} \\ P^{HS} &= T_{SH} P^{SS} T_{SH} \end{aligned}$$

There are three terms in the right side of this expression: the first one depends on the eye-in-head rotational velocity,

$\boldsymbol{\Omega}_{\mathbf{EH}}$. The second one depends on the head-on-shoulder rotational velocity, $\boldsymbol{\Omega}_{\mathbf{HS}}$. The third term is related to the target velocity in the shoulder-centered shoulder-fixed reference frame, $\dot{P^{SS}}$. When the eye-head-shoulder configuration is static during a trial, only the last term remains. In this case, the only change in velocity is due to the different orientations of the reference frames.

The first step for computing the retinal position and motion was to express the target position and velocity in a reference frame centered on the eye and fixed to the eye, $P^{EE}(t)$ and $\dot{P^{EE}}$. Then the second step is to compute the projection of the target position onto the retina and the resulting velocity of this projection, which is the retinal position and velocity available for further processing in the brain. Computationally, it amounts to project $P^{EE} = 1 + \epsilon\, \mathbf{P^{EE}}$ onto a sphere of radius r_{eye} (see **Figure 2A**). For simplicity, we assume $r_{eye} = 1$ (it is just a scaling factor).

$$projP^{EE} = 1 + \epsilon\, \mathbf{projP^{EE}}$$

where

$$\mathbf{projP^{EE}} = \frac{\mathbf{P^{EE}}}{\|\mathbf{P^{EE}}\|} = \frac{\mathbf{P^{EE}}}{d_P}$$

where $d_P \triangleq \|\mathbf{P^{EE}}\|$ is the distance of the target (in eye-centered coordinates). The velocity of the projection is computed as follows:

$$\dot{\mathbf{projP}}^{\mathbf{EE}} = \frac{\dot{\mathbf{P}}^{\dot{\mathbf{EE}}}}{d_P} - \frac{\dot{d}_P \mathbf{P^{EE}}}{d_P^2}$$

The resulting dual quaternion is $\dot{projP}^{EE} = \epsilon\, \dot{\mathbf{projP}}^{\mathbf{EE}}$. Note that if the motion of the target is spherical with the eye as center (isodistance trajectory), then $\dot{d}_P = 0$, and the expression simplifies to $\dot{\mathbf{projP}}^{\mathbf{EE}} = \frac{\dot{\mathbf{P}}^{\dot{\mathbf{EE}}}}{d_P}$. However, most of the time, the distance is not constant, for example, if the target is moving in a frontoparallel plane. $\dot{d}_P$ can be expressed as (see **Appendix F**):

$$\dot{d}_P = \frac{\mathbf{P^{EE}} \cdot \dot{\mathbf{P}}^{\dot{\mathbf{EE}}}}{d_P}$$

In this first part, we have computed the retinal target projection vector in eye-centered eye-fixed coordinates from the knowledge of target motion in spatial coordinates. This transformation is useful because we do not know the projection vector in retinal coordinates in advance. Therefore, from an experimental or simulation point of view, this transformation is important. Moreover the above transformation could be useful for the brain if the target is also known to the brain in spatial (or also head) coordinates (for example: auditory target, proprioceptive target in addition to the visual information about the target). Indeed, multisensory integration is carried out using multiple reference frames (McGuire and Sabes, 2009) and therefore auditory (head coordinates) information should be expressed in eye coordinates. This transformation could also be useful if the brain was to implement a forward model, predicting the sensory consequences of the arm motor command, in retinal coordinates.

Now we move to the inverse transformation, which computes the target motion in spatial (or shoulder) coordinates, with the retinal position and velocity as input. This transformation must be carried out by the brain in order to specify a spatially accurate motor plan for the arm (see Blohm and Crawford, 2007; Leclercq et al., 2012).

3.1.2. Computing the spatial position and velocity

Now we describe the model with $projP^{EE}$ and $\dot{projP}^{EE}$ as inputs, and P^{SS} and $\dot{P^{SS}}$ as outputs. These outputs would be used to specify the motor plan (direction, velocity) of an arm tracking movement for example (Leclercq et al., 2012). Using this model, predictions about the motor plan can therefore be easily computed under different hypotheses of incomplete transformation. These hypotheses are made by asking whether extra-retinal signals (3D eye and head positions, 3D eye and head velocities) are available to the brain and whether they are biased or not, and whether they are highly or not much variable (in the framework of bayesian estimation for example).

We start the complete transformation by reversing the above described model. From $projP^{EE}$ and $\dot{projP}^{EE}$, we first compute P^{EE} and $\dot{P^{EE}}$ (see **Figure 2B**). Theoretically,

$$\mathbf{P^{EE}} = d_P\, \mathbf{projP^{EE}}$$

We observe that d_P is necessary to reconstruct P^{EE}. It is interesting to consider how the brain estimates d_P. Most of the time, our vision is binocular and d_P can be estimated with the use of binocular cues (see Blohm et al., 2008). However, vision is sometimes monocular (e.g., Leclercq et al. (2012) patched one eye in order to have only monocular vision). In this case, no binocular clues are available to estimate target depth. But monocular cues and *a priori* information also help to have an estimation of the target depth, but they are quite reduced for point-like target movements in complete darkness (see Leclercq et al., 2012). A paradigm of

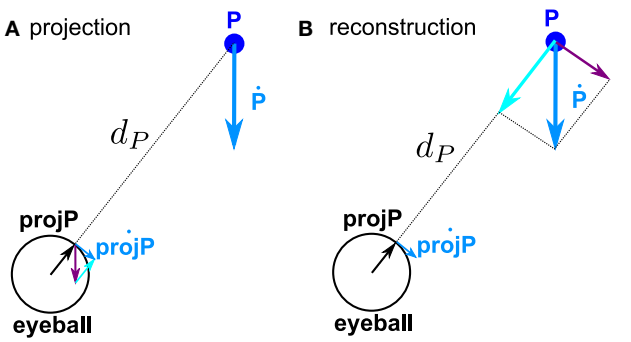

FIGURE 2 | Geometry of the velocity projection (A) and velocity reconstruction (B). (A) Projection of P position and velocity onto the eye (actually we plot the reversed retinal projection and velocity projection, to be directly comparable to the spatial position and velocity). References to the reference frame are omitted on the figure for the sake of clarity. The two components which add to obtain $\dot{\mathbf{projP}}^{\mathbf{EE}}$ are represented : $\dot{\mathbf{projP}}^{\mathbf{EE}} = \frac{\dot{\mathbf{P}}^{\dot{\mathbf{EE}}}}{d_P} - \frac{\dot{d}_P \mathbf{P^{EE}}}{d_P^2}$. (**B**) Reconstruction of P position and velocity. The two components which add to obtain $\dot{\mathbf{P}}^{\mathbf{EE}}$ are represented: $\dot{\mathbf{P}}^{\mathbf{EE}} = \dot{d}_P\, \mathbf{projP^{EE}} + \dot{d}_P \mathbf{projP^{EE}}$.

tracking in depth would be interesting to test, in monocular and binocular situations. That would allow to differentiate between monocular and binocular clues in order to estimate depth in the context of arm movements.

Theoretically, we also have:

$$\dot{\mathbf{P}^{\mathbf{EE}}} = d_P \, \dot{\mathbf{proj}}\mathbf{P}^{\mathbf{EE}} + \dot{d}_P \mathbf{proj}\mathbf{P}^{\mathbf{EE}}$$

thus $\dot{d}_P$ also has to be estimated for the estimation of the target velocity in space.

To compute the target position in spatial coordinates, we invert Equation (16), using the fact that the kinematic dual quaternions are unitary and therefore, for any unitary dual quaternion A, we have $A^{-1} = A^*$. Applying that, we have:

$$P^{\mathrm{SS}} = T_{\mathrm{HS}} R^*_{\mathrm{HS}} T_{\mathrm{EH}} R^*_{\mathrm{EH}} P^{\mathrm{EE}} R_{\mathrm{EH}} T_{\mathrm{EH}} R_{\mathrm{HS}} T_{\mathrm{HS}}$$

where $T_{\mathrm{EH}} = T^*_{\mathrm{HE}} = 1 - \epsilon \frac{\mathbf{t_{he}}}{2}$ and $T_{\mathrm{HS}} = T^*_{\mathrm{SH}} = 1 - \epsilon \frac{\mathbf{t_{sh}}}{2}$

For the target velocity in spatial coordinates, we inverse the relationship (17) and obtain:

$$\begin{aligned} \dot{P^{\mathrm{SS}}} = & \frac{1}{2}\left(P^{\mathrm{HS}} \mathbf{\Omega_{HS}} - \mathbf{\Omega_{HS}} P^{\mathrm{HS}}\right) \\ & + \frac{1}{2} R^*_{\mathrm{HS}} \left(P^{\mathrm{EH}} \mathbf{\Omega_{EH}} - \mathbf{\Omega_{EH}} P^{\mathrm{EH}}\right) R_{\mathrm{HS}} \\ & + R^*_{\mathrm{HS}} R^*_{\mathrm{EH}} \dot{P^{\mathrm{EE}}} R_{\mathrm{EH}} R_{\mathrm{HS}} \end{aligned}$$

These two transformations are implemented by the brain for the planning of spatially accurate movements, as it has been shown by Blohm and Crawford (2007) and Leclercq et al. (2012). These transformations are useful for the experimenter, in order to estimate the retinal position and motion of a target, and also to make different predictions on the motor plan generated by the brain, depending on hypotheses made about the signals available to the brain (accuracy, precision). However, we believe that dual quaternions are just a tool to easily model this transformation, and they are most likely not used by the brain as a way to implement these transformations. Most likely, these transformations are implemented in a distributed way by a neural network. Blohm et al. (2009) and Blohm (2012) show how these transformations could be achieved by the brain in a distributed way using a biologically inspired artificial neural network. Moreover they relate the properties of the artificial neurons to those of real neurons. Let us mention that dual quaternions are used in these theoretical studies to generate a huge number of input–output pairs (retinal motion as input and spatial motion as output).

3.2. TWO-LINK ARM MOVEMENTS: FORWARD AND INVERSE KINEMATICS FOR END-EFFECTOR POSITION

We will now consider another example of active movements: 3D arm movements. In section 3.2, we focus on the end-effector trajectory of a two-link arm, describing its forward kinematics and the inverse kinematics using the dual quaternion formalism applied to point transformations. In section 3.3, we consider a three-link arm holding a tool (a screwdriver for example) to emphasize the end-effector orientation importance in several everyday life situations. The dual quaternion formalism is again used in order to model the forward and inverse kinematics for the orientation and position of the end-effector, but it is also applied to lines, in order to deal with the orientation transformation.

3.2.1. Forward kinematics for end-effector position

In the following, we consider the forward kinematics of the end-effector position of a two-link arm (see **Figure 3**). The forward kinematics consists in computing the end-effector position (and sometimes orientation) from the knowledge of the joints kinematics (rotation angles and axes, joint velocities). The advantage of dual quaternions to represent such transformations is the compactness and geometrical significance to express the joint kinematics.

Assuming a length of a (resp. b) for the upper arm (resp. lower arm), the position P_{BS} of the end-effector P (B for body-fixed coordinates and S for **s**houlder-centered coordinates) can be expressed as:

$$P_{\mathrm{BS}} = R^*_{\mathrm{UB}} T_{\mathrm{ES}} R^*_{\mathrm{LU}} P_{\mathrm{LE}} R_{\mathrm{LU}} T_{\mathrm{ES}} R_{\mathrm{UB}} \tag{18}$$

where S stands for shoulder, U for upper arm, E for elbow and L for lower arm. R_{UB} represents the upper arm rotation around the shoulder compared to a reference position (rotation angle: $\varphi = 0$). R_{LU} represents the lower arm rotation around the elbow compared to a reference position ($\theta = 0$). $T_{\mathrm{ES}} = 1 + \epsilon\frac{1}{2}(a\ 0\ 0)^T$ represents translation dual quaternion associated with the offset between the elbow and the shoulder, in shoulder-centered upper-arm fixed coordinates. P_{LE} is the (fixed) position of point P in lower-arm fixed elbow-centered coordinates:

$$P_{\mathrm{LE}} = 1 + \epsilon \mathbf{P_{LE}} = 1 + \epsilon(b\ 0\ 0)^T$$

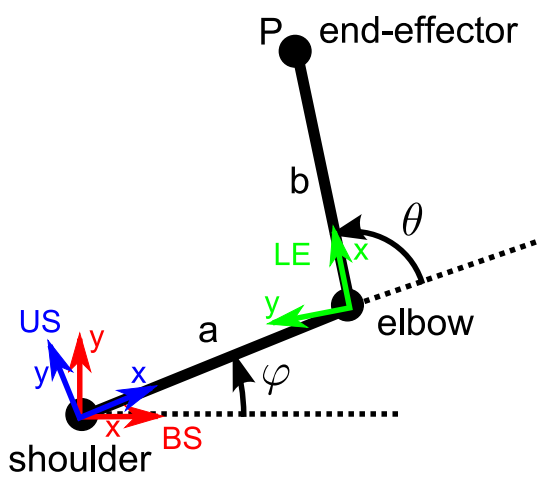

FIGURE 3 | Two-link arm top view. The shoulder insertion point, elbow and end-effector location are represented. The different reference frames are also represented: body-fixed shoulder-centered (BS, in red), upper arm -fixed shoulder-centered (US, in blue), and the lower arm -fixed elbow centered (LE, in green). The rotation angles of the shoulder and elbow are respectively denoted φ and θ.

Note that we can represent the forward kinematics of the end-effector if we know the parameters of the shoulder and elbow rotations (for example, angle and rotation axis), whatever the degrees of freedom.

We can also express the velocity $\dot{P}_{BS}$ of the end-effector P:

$$\dot{P}_{BS} = \frac{1}{2}\left(P_{BS}\boldsymbol{\Omega_{UB}} - \boldsymbol{\Omega_{UB}}P_{BS}\right) + R^*_{UB}\frac{P_{UE}\boldsymbol{\Omega_{LU}} - \boldsymbol{\Omega_{LU}}P_{UE}}{2}R_{UB} \quad (19)$$

where we obviously assume that $\dot{\mathbf{P}}_{\mathbf{LE}} = \mathbf{0}$. $\boldsymbol{\Omega}_{\mathbf{UB}}$ and $\boldsymbol{\Omega}_{\mathbf{LU}}$ represent the rotational velocities of the upper arm around the shoulder and of the lower arm around the elbow.

3.2.2. Inverse kinematics from end-effector position

Estimating the joint kinematic parameters from the knowledge of the end-effector position and velocity (and sometimes the orientation, see next) is a much more difficult task in general. This process is called inverse kinematics. It can be difficult for several reasons. First, we could solve the inverse kinematics equations (e.g., Equation 19) for the joint kinematic parameters, but theses equations are highly non-linear and no general method exists to solve this problem. For a particular case, it is sometimes possible and advantageous to compute analytical solutions, but it can not always be done. Numerical methods are also widely used to solve this problem, and in the following we will use this approach, which allows us to develop one general method for several different problems. Then, the kinematic system is usually redundant (like the human arm), meaning that there is an infinity of joint configurations that can achieve the prescribed end-effector kinematics. Finally, there are some geometrical configurations in which there are singularities and which tend to complicate the resolution of the inverse kinematics problem. This is a well studied problem in the field of robotics (Klein and Huang, 1983; Cheng and Gupta, 1991; Wang and Chen, 1991; Sciavicco and Siciliano, 1996; Tolani, 2000). Here, we take the two-link arm as an example and develop a methodology from our dual quaternion formalism to compute the inverse kinematics numerically.

The inverse kinematics problem is complicated, especially because the degree of freedom (dof) exceeds the dimension of the end-effector motion (this is called redundancy). For the two-link arm, we consider four degrees of freedom (three at the shoulder and one at the elbow), while the end-effector motion is only 3D. Therefore, there is an infinity of solutions to the problem.

First, we simplify the problem: we assume that the two-link arm moves in the horizontal plane, which reduces the joint dof to 2, yielding a well-posed problem for inverse kinematics. In this case, the velocity dual quaternion representing the 2D velocity of point P may be written as (see details in **Appendix G.1**):

$$\dot{P}_{BS} = \underbrace{\left(\frac{1}{2}\left(P_{BS}\mathbf{n} - \mathbf{n}P_{BS}\right) \quad R^*_{UB}\frac{P_{UE}\mathbf{n} - \mathbf{n}P_{UE}}{2}R_{UB}\right)}_{J_{DQ}}\left(\dot{\varphi} \quad \dot{\theta}\right)^T \quad (20)$$

where $\mathbf{n}$ is the rotation axis of the shoulder and the elbow (since we only consider planar motions) and where J_{DQ} is an array composed of two point velocity dual quaternions (from an implementation perspective, J_{DQ} is an 8×2 array, see section 2.6). We can build the 2×2 jacobian matrix J by extracting the 2D vector part of the velocity dual quaternion composing each column of J_{DQ} (see Equation 20):

$$J = \left(\left[\frac{1}{2}\left(P_{BS}\mathbf{n} - \mathbf{n}P_{BS}\right)\right]_{2Dvec} \quad \left[R^*_{UB}\frac{P_{UE}\mathbf{n} - \mathbf{n}P_{UE}}{2}R_{UB}\right]_{2Dvec}\right)$$

The dual quaternion equation (20) can then be expressed in a matrix form:

$$\dot{\mathbf{P}}_{\mathbf{BS}} = J\left(\dot{\varphi} \quad \dot{\theta}\right)^T$$

where $\dot{\mathbf{P}}_{\mathbf{BS}}$ is the 2D velocity vector component of the velocity dual quaternion $\dot{P}_{BS}$. We then compute $\dot{\varphi}$ and $\dot{\theta}$ as a function of $\dot{\mathbf{P}}_{\mathbf{BS}}$ to solve the inverse kinematics problem, assuming we know the current position of the joints, θ and φ, since the jacobian matrix J depends on these parameters: $J = J(\theta, \varphi)$. Indeed, J depends on P_{BS}, and P_{BS} depends on the rotation dual quaternions R_{UB} and R_{LU} (see Equation 17). Since θ is the rotation angle of R_{LU} and φ is the rotation angle of R_{UB}, it shows that J depends on θ and φ. Also, R_{UB} and P_{UE} depend on φ and appear in the expression of J.

However, we can solve for $\dot{\varphi}$ and $\dot{\theta}$ by inverting J only if J is invertible. It is the case in most geometrical configurations of the two-link arm (indeed, J depends on φ and θ, so does the rank of J), except when $\theta = 0$ (or $\theta = 180°$), which represents a situation where the upper and lower arms are aligned. In this case, J is not invertible (its rank is equal to 1), and we can not apply the above method. This situation can correspond to two cases. If $\dot{\mathbf{P}}_{\mathbf{BS}}$ has a component parallel to the aligned arm, then there are no solutions for $\dot{\varphi}$ and $\dot{\theta}$. If $\dot{\mathbf{P}}_{\mathbf{BS}}$ is orthogonal to the aligned arm, then there exists an infinity of solutions (in which case we can express the general form of the solution using the pseudo-inverse formalism, see later in the manuscript).

We notice that the differential velocity relationship linking the joint velocities and end-effector velocity is linear if we assume that θ and φ are known, and that will be the case in the numerical approach we use to solve the inverse kinematics problem. In order to solve the inverse kinematics problem for the joint positions, we can numerically integrate the joint velocities across time, knowing the joint positions at time t_0 (an initial condition is needed). The simplest numerical integration is the Euler method:

$$\begin{aligned}\theta(t + \Delta t) &= \theta(t) + \dot{\theta}(t)\Delta t \\ \varphi(t + \Delta t) &= \varphi(t) + \dot{\varphi}(t)\Delta t\end{aligned} \quad (21)$$

where Δt is the integration step and should not be too large to achieve reasonable accuracy. Then, the new rotation dual quaternions can be updated as a function of $\theta(t + \Delta t)$ and $\varphi(t + \Delta t)$ and we can iterate this process.

Now we want to consider movements in 3D space, not restricted to the plane. In this context, the shoulder has three-dof and the elbow has one-dof. It is clear that there are too many dof compared to the 3D motion of the end-effector, and intuitively many shoulder-elbow configurations will lead to the same end-effector position and velocity. In the following we explain one

approach which can be taken to obtain one specific solution for the joints, as well as a general formula to express the set of all possible solutions.

The idea is to start with the dual quaternion expression of the velocity of the end-effector (e.g., Equation 19) and rewrite it in a matrix expression. We move from the dual quaternion representation to the matrix notation because we use tools from the matrix algebra (e.g., the pseudo-inverse matrix) to derive the inverse kinematics results (see below).

The relationship between the end-effector velocity $\dot{P}_{\mathrm{BS}}$ and the rotational velocities of the shoulder, $\mathbf{\Omega_{UB}}$, and the elbow, $\mathbf{\Omega_{LU}}$, is described by Equation (19). Using the fact that: $\frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}}P_{\mathrm{BS}}\right) = -\epsilon\left(\vec{P}_{\mathrm{BS}} \wedge \vec{\Omega}_{\mathrm{UB}}\right)$ (see **Appendix G.2**) we can show that Equation (19) can be expressed as a linear matrix expression (see **Appendix G.3** for the derivation):

$$\underbrace{-\begin{pmatrix}\tilde{A}_{\mathbf{P_{BS}}} & R^{M}_{\mathrm{UB}}\tilde{A}_{\mathbf{P_{UE}}}\end{pmatrix}}_{J}\begin{pmatrix}\mathbf{\Omega_{UB}}\\ \mathbf{\Omega_{LU}}\end{pmatrix} = \dot{\mathbf{P}}_{\mathbf{BS}} \tag{22}$$

where $\dot{\mathbf{P}}_{\mathbf{BS}}$ is the dual bivector part of $\dot{P}_{\mathrm{BS}}$, R^{M}_{UB} is the rotation matrix associated with the rotation dual quaternion R_{UB} [see Equation (D.2) in **Appendix D** for how to compute this rotation matrix] and $\tilde{A}_{\mathbf{P_{BS}}}$ is the anti-symmetric matrix of rank 2 associated with the cross product: if $\vec{v} = \vec{a} \times \vec{b}$, it can also be written $\vec{v} = \tilde{A}_{\mathbf{a}}\vec{b}$ where:

$$\tilde{A}_{\mathbf{a}} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

where $\mathbf{a} = (a_1\ a_2\ a_3)$. Note that this matrix has always rank 2 since for an equation in $\vec{x}$: $\vec{a} \times \vec{x} = \vec{v}$, the component of $\vec{x}$ parallel to $\vec{a}$ can be arbitrary. All the solutions $\vec{x}$ lie on a line.

Coming back to Equation (22), we see that the matrix J is of size 3×6 and we want to solve for the unknowns $\mathbf{\Omega_{UB}}$ and $\mathbf{\Omega_{LU}}$. Since we consider one-dof at the elbow, the vector $\mathbf{\Omega_{LU}}$ must be aligned with a specific rotation axis $\mathbf{n}_{\mathrm{elbow}} = [0\ 0\ 1]$ (in upper arm fixed coordinates) since we assume that the elbow rotates around an axis which is orthogonal to both the lower and upper arms (see also **Figure 3** to see why this axis is along the z-axis) . This constraint can be written as $\vec{n}_{\mathrm{elbow}} \times \vec{\Omega}_{\mathrm{LU}} = \vec{0}$. Therefore, we add this constraint to the kinematics equation (22) to obtain:

$$\underbrace{\begin{pmatrix}-\tilde{A}_{\mathbf{P_{BS}}} & -R^{M}_{\mathrm{UB}}\tilde{A}_{\mathbf{P_{UE}}} \\ \mathbf{0} & \tilde{A}_{\mathbf{n}_{\mathrm{elbow}}}\end{pmatrix}}_{J}\begin{pmatrix}\mathbf{\Omega_{UB}}\\ \mathbf{\Omega_{LU}}\end{pmatrix} = \begin{pmatrix}\dot{\mathbf{P}}_{\mathbf{BS}}\\ \mathbf{0}\end{pmatrix} \tag{23}$$

Now this modified matrix J has generally rank 5 since $\tilde{A}_{\mathbf{n}_{\mathrm{elbow}}}$ has rank 2 (but the rank can be even lower in certain geometrical configurations of the two-link arm, as explained previously with the two-dof two-links arm). Indeed, because we can rotate the two-link arm system around an axis linking the shoulder and the end-effector, without changing the end-effector position and velocity, there is an infinity of solutions ($\mathbf{\Omega_{UB}}$, $\mathbf{\Omega_{LU}}$) which lie in a one-dimensional manifold, which is mathematically defined as the kernel of J, i.e., the set $\mathrm{Ker}(J) = \{\mathbf{x} \in \mathbb{R}^6 \text{ such that } J\mathbf{x} = \mathbf{0}\}$. Therefore, since there is an infinity of solutions, we need some criterion that may be optimized to choose one optimal solution (according to this criterion), while Equation (23) is a constraint for the optimization problem.

The matrix J^{-1} is not defined and thus we choose to use a generalized inverse matrix (which exists for any matrix): the pseudo-inverse [see Klein and Huang (1983) for the use of pseudo-inverse in inverse kinematics], also called the Moore–Penrose inverse. We will denote it by J^{+}. Therefore, one solution of the redundant system given by Equation (23) is given by:

$$\begin{pmatrix}\mathbf{\Omega_{UB}}\\ \mathbf{\Omega_{LU}}\end{pmatrix} = J^{+}\dot{\mathbf{P}}_{\mathbf{BS}} \tag{24}$$

This solution is actually the solution with minimal norm (Klein and Huang, 1983; Sciavicco and Siciliano, 1996), which minimizes the joint velocities in the context of the four-dof two-link arm. The general solution can also be expressed (see Sciavicco and Siciliano, 1996):

$$\begin{pmatrix}\mathbf{\Omega_{UB}}\\ \mathbf{\Omega_{LU}}\end{pmatrix} = J^{+}\dot{\mathbf{P}}_{\mathbf{BS}} + \left(I - J^{+}J\right)\mathbf{w}$$

where $\mathbf{w}$ can be any vector in $\mathbb{R}^6$, and I is the identity matrix of order 6 (for this example). In practice, Equation (24) requires that we explicitly compute the pseudo-inverse J^{+}. The interested reader can refer to **Appendix G.4** to see how this can be done.

Once we have a solution for the joint velocities $x = \begin{pmatrix}\mathbf{\Omega_{UB}}\\ \mathbf{\Omega_{LU}}\end{pmatrix}$, we can numerically integrate the joint velocities across time, knowing the joints positions at time t_0. Several methods for numerical integration exist (see for example, Atkinson, 1989) and are used in the context of inverse kinematics (see Cheng and Gupta, 1991). Here, we will use the simple Euler method to update the rotation quaternions R_{UB} and R_{LU}. It is a bit more tricky than in the simple two-dof two-link arm (see Equation 21). Indeed, given that $\dot{R} = \frac{1}{2}R\mathbf{\Omega}$, we can update the rotation dual quaternion as:

$$R(t + \Delta t) = R(t) + \dot{R}(t)\Delta t \tag{25}$$

but the major problem is that in general $R(t + \Delta t)$ is not a rotation quaternion anymore, and we have to normalize it by the norm of $R(t + \Delta t)$ to ensure it is a rotation dual quaternion again. From a computational perspective, Funda et al. (1990) showed that the normalization operation is carried out much faster with quaternions than with rotation matrices, which is one of the advantages of using quaternions over rotation matrices. One alternative for the computation is the following. In angular vector notation, the magnitude of the vector represents the rotation angle while the normalized unit vector represents the rotation axis. Using this fact, the rotational displacement between time t and $t + \Delta t$ is characterized by the angular vector $\mathbf{\Omega}\Delta t$ which is of the form $\theta\mathbf{n}$. Then, this angular vector can be expressed as a rotation dual quaternion (see Equation 3), ΔR. Then,

$$R(t + \Delta t) = R(t)\Delta R$$

However, by using such a numerical integration scheme, errors arise since the numerical integration is not perfect. These errors propagate from one iteration to the other and the reconstructed end-effector location (using the forward kinematic model, see Equation 18) will drift from the real end-effector position (see Sciavicco and Siciliano, 1996). In order to avoid this problem, we take the position error between the reconstructed end-effector position, $\hat{\mathbf{P}}_{\mathbf{BS}}$, and the real (or desired) end-effector position, $\mathbf{P}_{\mathbf{BS}}$, into account. Actually, we apply the correction scheme described in Sciavicco and Siciliano (1996). The vector error at time t is:

$$\begin{aligned}\mathbf{e}(t) &= \mathbf{P}_{\mathbf{BS}}(t) - \hat{\mathbf{P}}_{\mathbf{BS}}(t) \\ &= \mathbf{P}_{\mathbf{BS}}(t) - \mathrm{FK}(R_{\mathrm{LU}}(t), R_{\mathrm{UB}}(t))\end{aligned}$$

where FK(.) is the forward kinematic function described by Equation (18). We take this error into account by adding a vector term proportional to the error in Equation (23):

$$J\begin{pmatrix}\boldsymbol{\Omega}_{\mathbf{UB}}(t)\\ \boldsymbol{\Omega}_{\mathbf{LU}}(t)\end{pmatrix} = \begin{pmatrix}\dot{\mathbf{P}}_{\mathbf{BS}}(t) + K\mathbf{e}(t)\\ \mathbf{0}\end{pmatrix} \tag{26}$$

where K is a positive definite (usually diagonal) matrix. We can for example choose K as the identity matrix multiplied by a factor k_1, which is tuned by the user. Then, we can simply compute a solution using the same pseudo-inverse technique as described above [see Equation (24) to this modified problem].

We described how we deal with the inverse kinematics of the end-effector position of a two-link arm but we can generalize this procedure to a n-link arm. The interested reader can refer to **Appendix G.5**.

3.2.3. Inverse kinematics: numerical simulation

Here we test the numerical method for inverse kinematics developed in section 3.2.2 on the four-dof two-link arm. **Figure 4** shows one particular example of inverse kinematics for this two-link arm. First, we choose an end-effector motion with a bell-shaped velocity profile and a straight line path in the 3D workspace (see **Figure 4B**), to mimic hand velocity profiles described in the neuroscience literature (Abend et al., 1982; Atkeson and Hollerbach, 1985). From the subject perspective, the hand is moving from a central position to the right, up and away from the shoulder. The initial configuration of the arm is represented in black (see **Figure 4A**). From the end-effector motion, we apply the inverse kinematics technique described in the previous section to estimate the joint rotations over time for the movement (**Figure 4C**).

The estimation accuracy was assessed by computing the norm of the position error, i.e., the difference between the true end-effector position and the estimated end-effector position. The end-effector position was estimated by using the forward kinematic model (see Equation 19) and the estimated joint angles (and therefore rotation quaternions). We can observe (**Figure 4D**) that the position error was always smaller than 0.02 cm. This position error depends on the value of k_1 (see Equation 26) that we use. For this simulation, we used $k_1 = 1000$. The larger k_1 the smaller is the position error, because we penalize the position error more with larger values of k_1. But k_1 can not be too large for discrete time systems. If k_1 is too large, the system will be unstable and the error will grow. The *threshold* value for k_1 depends on the simulation step Δt (see **Appendix G.6** for details). For our simulation, we used $\Delta t = 1$ ms and therefore we could not choose a value for k_1 larger than 2000.

We tested several directions in space for the end-effector motions and our inverse kinematics algorithm was successful in every case, inferring joint angles for the elbow and shoulder. Note that this method does not necessarily reproduce the joints angles observed in human subjects but it yields the solution with the smallest joint angle rotations.

3.3. THREE-LINK ARM MOVEMENTS: FORWARD AND INVERSE KINEMATICS FOR END-EFFECTOR POSITION AND ORIENTATION

In this section, we describe the forward and inverse kinematics for the end-effector position and orientation of a three-link arm. The main difference compared to section 3.2 is that we perform line transformations (using dual quaternions) to transform the orientation.

For this section, we decided to provide an example where we directly use screw motion to model the joints (shoulder, elbow, and wrist) movements, but we could have kept the same approach as in section 3.2 by alternating rotations and translation (offsets). Similarly, we could have used screw motion dual quaternions as well in section 3.2 for the two-link arm example.

3.3.1. Forward kinematics for end-effector position and orientation

Here, we describe a seven-dof arm with three links (see **Figure 5**): the upper and lower arms as well as the wrist. We consider a seven-dof joint model: three for the shoulder, one for the elbow, and three for the wrist. Up to here, we could just apply the methodology described above in section 3.2. However, now we consider an example where the end-effector *orientation* matters. For example, **Figure 5** shows that the hand holds a screwdriver, and we are interested in the location and orientation of this end-effector.

In order to describe the end-effector (the screwdriver) orientation (in body-fixed coordinates), we now use the formalism of line dual quaternions (see section 2.3.2). Let $L_0 = \mathbf{n}_0 + \epsilon(\vec{r}_0 \wedge \vec{n}_0)$ be the line dual quaternion describing the reference line passing through the screwdriver in its reference configuration [see the gray configuration in **Figure 5**: the arm is fully extended, $\varphi = \theta = \alpha = 0°$ and $\mathbf{n}_0 = (0\ 1\ 0)^T$ and $\mathbf{r}_0 = (a+b+c\ 0\ 0)^T$]. The forward kinematics describing the end-effector line position, $L(t) = \mathbf{n}(t) + \epsilon\mathbf{m}(t)$, is described by line transformations, using the tools described in the previous sections (again, here we use screw motions but we could have used rotations and translations offsets):

$$\begin{aligned}L(t) &= \left(S^*_{\mathrm{UB}}S^*_{\mathrm{LU}}S^*_{\mathrm{HL}}\right) L_0(t) \underbrace{\left(S_{\mathrm{HL}}S_{\mathrm{LU}}S_{\mathrm{UB}}\right)}_{S_{\mathrm{tot}}} \\ &= S^*_{\mathrm{tot}}L_0S_{\mathrm{tot}}\end{aligned} \tag{27}$$

where:

- $S_{\mathrm{UB}} = R_{\mathrm{UB}} = \cos(\varphi/2) + \mathbf{n_{UB}}\sin(\varphi/2)$ is a pure rotation quaternion describing the 3D rotation of the upper arm around the shoulder.

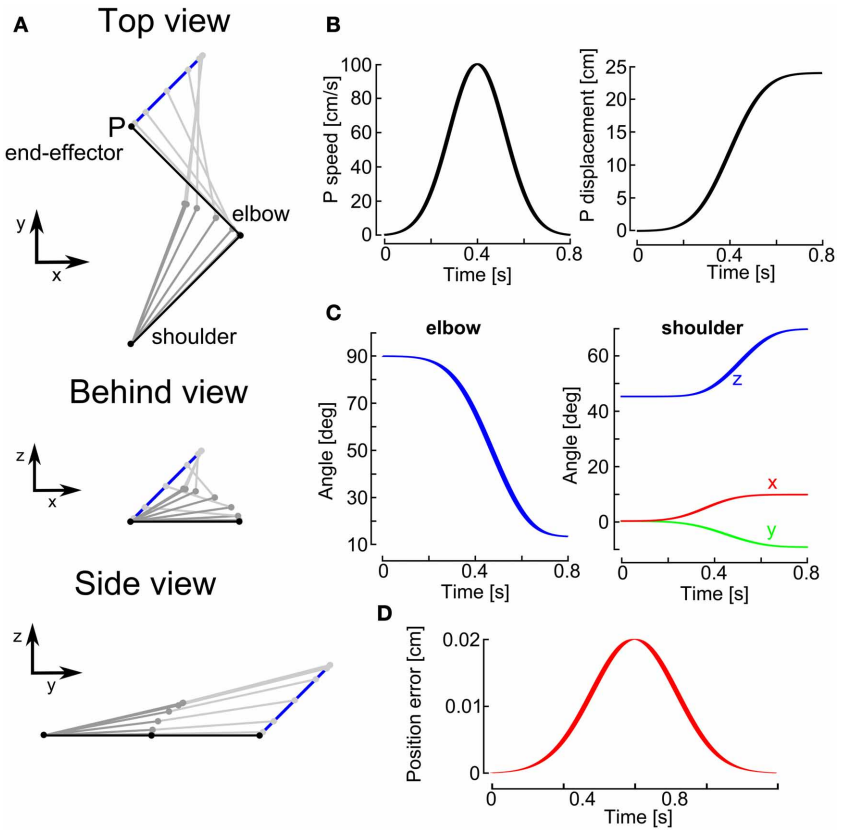

FIGURE 4 | Numerical simulation example of inverse kinematics of the four-dof two-link arm. For this simulation, we used $k_0 = 1000$ and $k_1 = 1$ (see **Appendix G.4**) and a simulation step of 1 ms duration. **(A)** Top view (top), behind view (middle), and side view (bottom) of the computed inverse kinematics for the joints. The end-effector (point P) motion in 3D space is described by the blue line. The initial two-link arm configuration is represented in black. Upper arm (dark gray) and lower arm (light gray) are represented at different instants during the end-effector motion. **(B)** The end-effector speed (left) and displacement (right) are described as a function of time: the movement is a typical bell-shaped speed profile lasting 800 ms and with a peak velocity of 100 cm/s. The end-effector displacement is about 25 cm. **(C)** The estimated elbow (left) and shoulder (right) rotation angles are represented as a function of time. For the shoulder, there are three components of rotation: around the x-axis (red line), the y-axis (green line), and the z-axis (blue line). **(D)** Representation of the position error, the norm of the difference between the estimated and actual end-effector position, as a function of time. The maximal error is about 0.02 cm.

- $S_{\mathrm{LU}} = T^*_{\mathrm{ES}} R_{\mathrm{LU}} T_{\mathrm{ES}}$ is a screw motion dual quaternion describing the 1D rotation of the lower arm around the elbow, whose rotation axis is offset from the origin (the shoulder):
  - $R_{\mathrm{LU}} = \cos(\theta/2) + \mathbf{n_{LU}} \sin(\theta/2)$ is a pure rotation quaternion describing the 3D rotation of the lower arm around the elbow.
  - $T_{\mathrm{ES}} = 1 + \epsilon \mathbf{t}_{\mathrm{es}}/2$ where $\mathbf{t}_{\mathrm{es}} = [a\ 0\ 0]^T$ is the offset between the elbow and the shoulder in the reference configuration.
- $S_{\mathrm{HL}} = T^*_{\mathrm{WS}} R_{\mathrm{HL}} T_{\mathrm{WS}}$ is a screw motion dual quaternion describing the 3D rotation of the hand around the wrist, whose rotation axis is offset from the origin (the shoulder):
  - $R_{\mathrm{HL}} = \cos(\alpha/2) + \mathbf{n_{HL}} \sin(\alpha/2)$ is a pure rotation quaternion describing the 3D rotation of the hand around the wrist.
  - $T_{\mathrm{WS}} = 1 + \epsilon \mathbf{t}_{\mathrm{ws}}/2$ where $\mathbf{t}_{\mathrm{ws}} = [a + b\ 0\ 0]^T$ is the offset between the wrist and the shoulder in the reference configuration.

We can also derive the expression for the end-effector line velocity, $\dot{L}(t)$, deriving the above expressions:

$$\begin{aligned}\dot{L}(t) = {} & S^*_{\mathrm{tot}} \dot{L}_0(t) S_{\mathrm{tot}} \\ & + S^*_{\mathrm{UB}} S^*_{\mathrm{LU}} \left(\dot{S}^*_{\mathrm{HL}} L_0 S_{\mathrm{HL}} + S^*_{\mathrm{HL}} L_0 \dot{S}_{\mathrm{HL}} \right) S_{\mathrm{LU}} S_{\mathrm{UB}} \\ & + S^*_{\mathrm{UB}} \left(\dot{S}^*_{\mathrm{LU}} L_w S_{\mathrm{LU}} + S^*_{\mathrm{LU}} L_w \dot{S}_{\mathrm{LU}} \right) S_{\mathrm{UB}} \\ & + \dot{S}^*_{\mathrm{UB}} L_e S_{\mathrm{UB}} + S^*_{\mathrm{UB}} L_e \dot{S}_{\mathrm{UB}}\end{aligned} \tag{28}$$

where

- $L_w = S^*_{\mathrm{HL}} L_0 S_{\mathrm{HL}}$ is the end-effector line position after applying the wrist rotation to the reference line L_0.
- $L_e = S^*_{\mathrm{LU}} L_w S_{\mathrm{LU}}$ is the end-effector line position after applying the elbow rotation to the line L_w.

Remembering the description of a screw motion velocity dual quaternion $\dot{M}$ (see Equation 13) in terms of the pure rotational

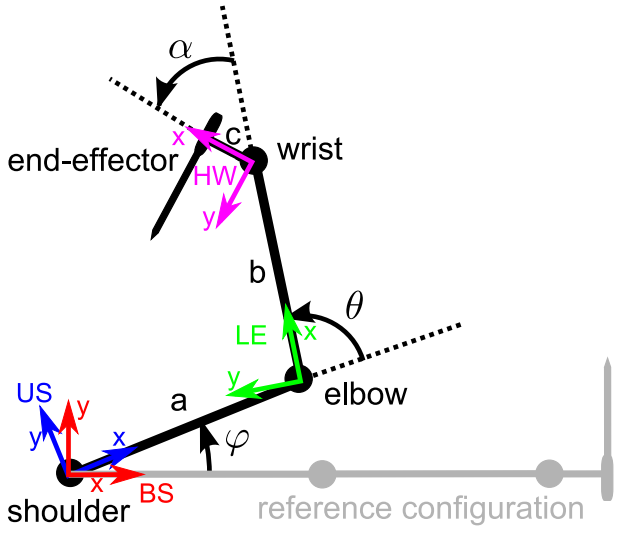

FIGURE 5 | seven-dof three-link arm top view. The shoulder insertion point, elbow and wrist are represented as well as the end-effector (a screwdriver) location and orientation. The different reference frames are also represented: body-fixed shoulder-centered (BS, in red), upper arm -fixed shoulder-centered (US, in blue), lower arm -fixed elbow centered (LE, in green), and hand-fixed wrist-centered (HW, in magenta). The rotation angles of the shoulder, elbow, and wrist are respectively denoted φ, θ, and α. The configuration in gray represents the reference configuration.

component, $\mathbf{\Omega}$, the offset velocity term $\dot{T}_L$ and the translational velocity term along the screw axis $\dot{T}$, we can express these screw motion velocities for the seven-dof three-link arm. For each of the screw motion dual quaternions considered here, only the rotational part is moving, in which case $\dot{M}$ simplifies to $\dot{M} = \frac{1}{2}TT_L^* R\mathbf{\Omega}T_L$. Therefore, it is quite easy to compute $\dot{L}(t) = \dot{\mathbf{n}}(t) + \epsilon\dot{\mathbf{m}}(t)$.

From the line forward kinematics for position and velocity, we can derive the orientation of the end-effector, $\mathbf{n}(t)$, as well as its rate of change, $\dot{\mathbf{n}}(t)$. The actual position and velocity of the end-effector position is more tricky to obtain. Indeed, we can retrieve the quantities $\mathbf{m}(t)$ and $\dot{\mathbf{m}}(t)$, but with these quantities, we can only infer the component of the end-effector position orthogonal to the line orientation. And, even if the end-effector position is known, we can not uniquely determine the end-effector velocity component along the line orientation (see also **Appendix B**). Therefore, in parallel to the line transformation applied to the end-effector line, a position transformation should be applied to the end-effector position. The transformation equations applied to the reference point dual quaternion $P_0 = 1 + \epsilon\mathbf{r}_0$ are similar to Equations (27) and (28) except that we use the conjugate definition $\tilde{S}^*$ instead of S^* for point transformation (as described in section 2.4).

3.3.2. Inverse kinematics from end-effector position and orientation

In the case of line transformations, we are also interested in estimating the joint kinematic parameters from the knowledge of the end-effector position and orientation as well as the velocity and orientation rate of change. The problems and the challenges are similar to what we saw before in section 3.2.2 except for the construction of the jacobian matrix J which links the joint-velocities to the end-effector velocity and orientation rate of change. For the interested reader, **Appendix H** describes in detail how to compute this jacobian matrix J for the three-link arm and also generalizes to the n-link arm. Then we can proceed similarly to what is shown in section 3.2.2 to solve the inverse kinematics problem.

4. DISCUSSION

In many applications of behavioral neuroscience or vision, there is a need to represent the 3D position and/or orientation of objects as well as their velocity. These objects may be external objects/stimuli that a human subject has to deal with (e.g., the position and orientation of a target object). For example, in order to catch the ball, a rugby player needs to estimate the 3D motion of the ball but also the time course of its orientation, since the ball is non-spherical. This object can also be a body part (e.g., the eye, the head, and the hand), whose position and/or orientation is of interest for the brain to monitor body movements. Dual quaternions provide a convenient compact and geometrically meaningful way to describe either position (through point dual quaternions) or position and orientation (through line dual quaternions). Moreover, dual quaternions provide a way to describe natural geometrical transformations like rotations, translations, or screw motions, and these geometrical transformations can easily be combined together and applied to points or lines. In this paper, we described the useful concepts of the dual quaternion geometric algebra and how dual quaternions can be used to model these transformations. We also described the dual quaternion formalism to cope with velocity: rotational velocity, screw motion velocity, point velocity, and line velocity. Then we applied these concepts to a few examples in behavioral neuroscience: the 3D eye-head-shoulder system reference frame transformation needed for the accurate planning of manual tracking movements. Another example was provided with the forward kinematics for the multi-link arm, either considering the end-effector (the hand) position alone, or considering the end-effector position and/or orientation. Finally we also derived the inverse kinematics of the same multi-link arm from the dual quaternion formalism. In these applications, we do not claim that the brain actually uses dual quaternions to implement these transformations. However, these complex transformations are easily expressed mathematically using dual quaternions, which helps the neuroscience researcher to make predictions, for a theoretical goal or just in designing an experiment.

A main advantage of dual quaternions is that we can combine several rotations and/or translations. Therefore, it is quite easy to compute and write the expression for complex systems. We used that advantage throughout our applications. Another advantage of using quaternions (or dual quaternions) to represent rotations is that it is an efficient way to parameterize rotations, without any singularity (Chou, 1992; Grassia, 1998), which occur when using the classical ways to parameterize rotations (Euler angles for instance). An important advantage of dual quaternions is the compactness in terms of memory requirements: we only need 8 elements to represent them, compared to 12 for

homogeneous matrices (Kim and Kumar, 1990; Funda and Paul, 1990a; Aspragathos and Dimitros, 1998; Daniilidis, 1999; Kavan et al., 2008).

Quaternions have been widely used to model 3D eye rotations (Tweed and Vilis, 1987; Hestenes, 1994a; Haslwanter, 1995; Crawford and Guitton, 1997; Bayro-Corrochano, 2003) while dual quaternions have been used less (see Bayro-Corrochano, 2003; Blohm and Crawford, 2007; Leclercq et al., 2012). Here, we described how the dual quaternion formalism can be used to model the multi-body kinematics, which can be useful for modeling purposes in applications like motor planning for eye and arm movements, 3D eye-head gaze shifts, 3D VOR, 3D updating, computing predictions of an inverse or forward internal model.

Dual quaternions have been commonly used in robotics and computer vision, for various purposes. Quaternion parametrization of rotations are used in graphics applications (Grassia, 1998; Tolani, 2000; Azariadis, 2001; Kavan et al., 2008; Ding et al., 2012). Tolani (2000) developed a real-time inverse kinematic technique for anthropomorphic limbs. In Ding et al. (2012), they develop an image processing technique which is built upon a so-called dual quaternion Fourier transform. According to Grassia (1998), quaternions are the best choice to interpolate 3D rotations. Indeed, when we interpolate element by element between two rotations quaternions, it interpolates on the geodesic (shortest) path onto the sphere (Shoemake, 1985; Dam et al., 1998). Dual quaternions are also used for the process of pose estimation, which consists in estimating the position and orientation of an object. In this context, dual quaternions allow to solve simultaneously for both the position and orientation components (Daniilidis, 1999; Bayro-Corrochano et al., 2000) in the context of robotics (hand-eye calibration). Pose estimation is also used with dual quaternions in computer vision (Walker et al., 1991; Phong et al., 1993; Torsello et al., 2011), but with a projective component in addition. In Goddard and Abidi (1998), they used the iterative extended Kalman filter for this purpose of pose estimation in order to deal with uncertainty. Forward kinematics equations for robotic manipulators have been derived with dual quaternions (Kim and Kumar, 1990) and compared with other methods (Aspragathos and Dimitros, 1998). Perez and McCarthy (2004) use dual quaternions for the synthesis of constrained robotic systems where several serial constrained (less than six-dof) manipulators are combined. Inverse kinematics techniques have been developed using dual quaternions for six-dof manipulators (Aydin and Kucuk, 2006; Sariyildiz and Temeltas, 2009) and compared to other methods (Sariyildiz et al., 2011). They are also used in the context of cooperative control of multiple manipulators: several robotic manipulators (Adorno et al., 2010) or a robotic manipulator interacting with a human arm (Adorno et al., 2011). In Pham et al. (2010), they develop a control law using dual quaternion to control simultaneously the position and orientation of a robotic manipulator.

Thus dual quaternions are powerful mathematical constructs that are widely used in robotics and computer vision, and could make important contributions to neuroscience research.

ACKNOWLEDGMENTS

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State Science Policy Office. The scientific responsibility rests with its author(s). This work was also supported by the Belgian Fonds National de la Recherche Scientifique, the Fondation pour la Recherche Scientifique Médicale, and internal research grants from the Université catholique de Louvain (Fonds Spéciaux de Recherche, Action de Recherche Concertée), The Botterell Foundation (Queen's University, Canada), NSERC (Canada), and the Ontario Research Fund (Canada).

REFERENCES

Abend, W., Bizzi, E., and Morasso, P. (1982). Human arm trajectory formation. *Brain* 105, 331–348.

Adorno, B., Fraisse, P., and Druon, S. (2010). "Dual position control strategies using the cooperative dual task-space framework," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*, (Taipei), 3955–3960.

Adorno, B. V., Bo, A. P. L., and Fraisse, P. (2011). "Interactive manipulation between a human and a humanoid: when robots control human arm motion," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Fransisco, CA), 4658–4663

Aspragathos, N. A., and Dimitros, J. K. (1998). A comparative study of three methods for robot kinematics. *IEEE Trans. Syst. Man Cyberne. B Cybern.* 28, 135–145.

Atkeson, C. G., and Hollerbach, J. M. (1985). Kinematic features of unrestrained vertical arm movements. *J. Neurosci.* 5, 2318–2330.

Atkinson, K. (1989). *An introduction to numerical analysis, 2nd Edn*. New York, NY: John Wiley Edition.

Aydin, Y., and Kucuk, S. (2006). "Quaternion based inverse kinematics for industrial robot manipulators with Euler Wrist," in *2006 IEEE International Conference on Mechatronics*, (Budapest), 581–586.

Azariadis, P. (2001). Computer graphics representation and transformation of geometric entities using dual unit vectors and line transformations. *Comput. Graph.* 25, 195–209.

Bayro-Corrochano, E. (2003). Modeling the 3D kinematics of the eye in the geometric algebra framework. *Pattern Recogn.* 36, 2993–3012.

Bayro-Corrochano, E., Daniilidis, K., and Sommer, G. (2000). Motor algebra for 3D kinematics: the case of the hand-eye calibration. *J. Math. Imaging Vis.* 13, 79–100.

Blohm, G. (2012). Simulating the cortical 3D visuomotor transformation of reach depth. *PLoS ONE* 7:e41241. doi: 10.1371/journal.pone.0041241

Blohm, G., and Crawford, J. D. (2007). Computations for geometrically accurate visually guided reaching in 3-D space. *J. Vis.* 7, 4.1–4.22.

Blohm, G., Keith, G. P., and Crawford, J. D. (2009). Decoding the cortical transformations for visually guided reaching in 3D space. *Cereb. Cortex* 19, 1372–1393.

Blohm, G., Khan, A. Z., Ren, L., Schreiber, K. M., and Crawford, J. D. (2008). Depth estimation from retinal disparity requires eye and head orientation signals. *J. Vis.* 8, 3.1–3.23.

Cheng, H., and Gupta, K. (1991). A study of robot inverse kinematics based upon the solution of differential equations. *J. Robot. Syst.* 8, 159–175.

Chou, J. (1992). Quaternion kinematic and dynamic differential equations. *IEEE Trans. Robot. Automat.* 8, 53–64.

Crawford, J. D., and Guitton, D. (1997). Visual-motor transformations required for accurate and kinematically correct saccades. *J. Neurophysiol.* 78, 1447–1467.

Dam, E., Koch, M., and Lillholm, M. (1998). *Quaternions, interpolation and animation*. Technical report, University of Copenhagen.

Daniilidis, K. (1999). Hand-eye calibration using dual quaternions. *Int. J. Robot. Res.* 18, 286–298.

Ding, Z., Yu, Y., Wang, B., and Zhang, L. (2012). An approach for visual attention based on biquaternion and its application for ship detection in multispectral imagery. *Neurocomputing* 76, 9–17.

Funda, J., and Paul, R. P. (1990a). A computational analysis of screw

transformations for robotics. *IEEE Trans. Robot. Automat.* 6, 348–356.

Funda, J., and Paul, R. P. (1990b). A computational analysis of screw transformations in robotics. *IEEE Trans. Robot.* 6, 348–356.

Funda, J., Taylor, R. H., and Paul, R. P. (1990). On homogeneous transforms, quaternions, and computational efficiency. *IEEE Trans. Robot. Automat.* 6, 382–388.

Goddard, J. S., and Abidi, M. A. (1998). "Pose and motion estimation using dual quaternion-based extended Kalman filtering," in *Proceedings of SPIE Conference on Three-Dimensional Image Capture and Applications*. Vol. 3313, (San Jose, CA), 189–200.

Grassia, F. (1998). Practical parameterization of rotations using the exponential map. *J. Graph. Tools* 3, 29–48.

Haslwanter, T. (1995). Mathematics of three-dimensional eye rotations. *Vis. Res.* 35, 1727–1739.

Hestenes, D. (1986). *New Foundations for Classical Mechanics*. Dordrecht: D. Reidel Publishing Company.

Hestenes, D. (1994a). Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Netw.* 7, 65–77.

Hestenes, D. (1994b). Invariant body kinematics: II. Reaching and neurogeometry. *Neural Netw.* 7, 79–88.

Hore, J., Watts, S., and Vilis, T. (1992). Constraints on arm position when pointing in three dimensions: Donders' law and the Fick gimbal strategy. *J. Neurophysiol.* 68, 374–383.

Kavan, L., Collins, S., Žára, J., and O'Sullivan, C. (2008). Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.* 27, 1–23.

Kim, J.-H., and Kumar, V. R. (1990). Kinematic of robot manipulators via line transformations. *J. Robot. Syst.* 7, 649–674.

Klein, C., and Huang, C. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans. Syst. Man Cybern.* SMC-13, 245–250.

Leclercq, G., Blohm, G., and Lefèvre, P. (2012). Accurate planning of manual tracking requires a 3D visuomotor transformation of velocity signals. *J. Vis.* 12, 1–21.

McGuire, L. M. M., and Sabes, P. N. (2009). Sensory transformations and the use of multiple reference frames for reach planning. *Nat. Neurosci.* 12, 1056–1061.

Perez, A., and McCarthy, J. M. (2004). Dual quaternion synthesis of constrained robotic systems. *J. Mech. Des.* 126, 425.

Pham, H., Perdereau, V., Adorno, B., and Fraisse, P. (2010). "Position and orientation control of robot manipulators using dual quaternion feedback," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*, (Taipei), 658–663.

Phong, T., Horaud, R., Yassine, A., and Pham, D. (1993). "Optimal estimation of object pose from a single perspective view," in *1993 (4th) International Conference on Computer Vision* (Berlin: IEEE Computer Society Press), 534–539.

Sariyildiz, E., Cakiray, E., and Temeltas, H. (2011). A comparative study of three inverse kinematic methods of serial industrial robot manipulators in the screw theory framework. *Int. J. Adv. Robot. Syst.* 8, 9–24.

Sariyildiz, E., and Temeltas, H. (2009). "Solution of inverse kinematic problem for serial robot using dual quaterninons and plücker coordinates," in *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, (Singapore), 338–343.

Sciavicco, L., and Siciliano, B. (1996). *Modeling and Control of Robot Manipulators*. New York, NY: McGraw-Hill.

Shadmehr, R., and Wise, S. P. (2005). *The Computational Neurobiology of Reaching and Pointing: A Foundation for Motor Learning. Computational Neuroscience.* Cambridge, MA: MIT Press.

Shoemake, K. (1985). Animating rotation with quaternion curves. *ACM SIGGRAPH Comput. Graph.* 19, 245–254.

Straumann, D., Haslwanter, T., Hepp-Reymond, M.-C., and Hepp, K. (1991). Listing's law for eye, head and arm movements and their synergistic control. *Exp. Brain Res.* 86, 209–215.

Tolani, D. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graph. Models* 62, 353–388.

Torsello, A., Rodola, E., and Albarelli, A. (2011). "Multiview registration via graph diffusion of dual quaternions," in *CVPR 2011 (IEEE)*, (Providence, RI), 2441–2448.

Tweed, D., and Vilis, T. (1987). Implications of rotational kinematics for the oculomotor system in three dimensions. *J. Neurophysiol.* 58, 832–849.

Walker, M. W., Shao, L., and Volz, R. A. (1991). Estimating 3-D location parameters using dual number quaternions. *CVGIP Image Underst.* 54, 358–367.

Wang, L., and Chen, C. (1991). A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *Robot. Automat. IEEE Trans.* 7, 489–499.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 31 July 2012; accepted: 28 January 2013; published online: 20 February 2013.

Citation: Leclercq G, Lefèvre P and Blohm G (2013) 3D kinematics using dual quaternions: theory and applications in neuroscience. Front. Behav. Neurosci. 7:7. doi: 10.3389/fnbeh.2013.00007

APPENDICES

A. MULTIPLICATION OF TWO QUATERNIONS: PROOF

The multiplication of two quaternions $A = a_0 + i\vec{a}$ and $B = b_0 + i\vec{b}$ is a quaternion $C = c_0 + i\vec{c}$. It is computed as follows:

$$\begin{aligned} C &= AB \\ &= (a_0 + i\vec{a})(b_0 + i\vec{b}) \\ &= a_0 b_0 + i(a_0\vec{b} + b_0\vec{a}) - \vec{a}\vec{b} \\ &= a_0 b_0 + i(a_0\vec{b} + b_0\vec{a}) - (\vec{a}\cdot\vec{b} + \vec{a}\wedge\vec{b}) \\ &= \underbrace{(a_0 b_0 - \vec{a}\cdot\vec{b})}_{c_0} + i\underbrace{(a_0\vec{b} + b_0\vec{a} - \vec{a}\times\vec{b})}_{\vec{c}} \end{aligned}$$

using the property $i^2 = -1$ and the expression of the geometric product of two vectors (see Equation 1).

B. PROOF THAT THE LINE DERIVATIVE $\dot{L}$ DOES NOT DEPEND ON THE CHOICE OF $\vec{P}$

In section 2.3.2, we saw that:

$$\dot{L} = \dot{\mathbf{n}} + \epsilon\dot{\mathbf{m}} \tag{B.1}$$

$$= \dot{\mathbf{n}} + \epsilon\left(\dot{\vec{p}}\wedge\vec{n} + \vec{p}\wedge\dot{\vec{n}}\right) \tag{B.2}$$

One could argue that this line derivative expression depends on the choice of $\vec{p}$ to describe a point on line L, and how it changes on this line. We prove in the following that it is not the case. Assume we choose a particular representation: $\vec{p_1}$ and $\dot{\vec{p}}_1$. Then the dual component of expression (B.2) is:

$$c_1 = \dot{\vec{p}}_1\wedge\vec{n} + \vec{p_1}\wedge\dot{\vec{n}}$$

Assume now that we choose another representation, $\vec{p_2}$ and $\dot{\vec{p}}_2$ and that they are linked to the first one in the following way:

$$\begin{aligned} \vec{p_2} &= \vec{p_1} + x\vec{n} \\ \dot{\vec{p}}_2 &= \dot{\vec{p}}_1 + \dot{x}\vec{n} + x\dot{\vec{n}} \end{aligned}$$

Then, the dual component of expression (B.2) is:

$$\begin{aligned} c_2 &= \dot{\vec{p}}_2\wedge\vec{n} + \vec{p}_2\wedge\dot{\vec{n}} \\ &= \left(\dot{\vec{p}}_1 + \dot{x}\vec{n} + x\dot{\vec{n}}\right)\wedge\vec{n} + \left(\vec{p_1} + x\vec{n}\right)\wedge\dot{\vec{n}} \\ &= \dot{\vec{p}}_1\wedge\vec{n} + \dot{x}\underbrace{\vec{n}\wedge\vec{n}}_{=\vec{0}} + x\dot{\vec{n}}\wedge\vec{n} + \vec{p_1}\wedge\dot{\vec{n}} + x\vec{n}\wedge\dot{\vec{n}} \\ &= \dot{\vec{p}}_1\wedge\vec{n} + \vec{p_1}\wedge\dot{\vec{n}} \end{aligned}$$

which proves that $c_2 = c_1$ and that the representations are indeed independent of the choice of $\vec{p}$.

C. DIVERSE TOOLS AND RESULTS

C.1. Shortest rotation between two unitary vectors

Let us consider two unitary vectors $\vec{a_1}$ and $\vec{b_1}$. Their quaternion representation (or dual part of their dual quaternion representation) are $A_1 = 0 + \mathbf{a_1}$ and $B_1 = 0 + \mathbf{b_1}$. We can write the following:

$$\begin{aligned} A_1 &= B_1(B_1^{-1}A_1) \\ &= B_1\left(B_1^* A_1\right) \end{aligned} \tag{C.1}$$

The expression $B_1^* A_1$ can be developed:

$$\begin{aligned} B_1^* A_1 &= -\mathbf{b_1}\mathbf{a_1} \\ &= \vec{b_1}\vec{a_1} \\ &= \vec{b_1}\cdot\vec{a_1} + \vec{b_1}\wedge\vec{a_1} \\ &= \cos\theta + \mathbf{n}\sin\theta \end{aligned} \tag{C.2}$$

where θ is the rotation angle from vector $\vec{b_1}$ toward vector $\vec{b_1}$, and $\mathbf{n} = \frac{\vec{b_1}\wedge\vec{a_1}}{\sin\theta}$ is the unitary rotation axis. Together, these two parameters describe the shortest rotation from vector $\vec{b_1}$ to vector $\vec{a_1}$.

This way of finding the shortest rotation between two unitary vectors can for instance be used to describe the 3D eye rotation obeying Listing's law. As all 3D eye orientations can be described as a rotation from a reference eye position, the Listing's law states that all the possible rotation axes lie in a plane, which is orthogonal to a specific reference eye position, called the primary position. Knowing the primary position and the current gaze position, it is easy to compute the rotation quaternion: $R_{\text{rot}} = \cos\theta + \mathbf{n}\sin\theta$ describing the rotation angle and rotation axis of the eye, using Equation (C.2).

C.2. Shortest screw motion between two lines

Similarly to what is shown in **Appendix C.1**, we can find the shortest screw motion between two lines $L_0 = \mathbf{n_0} + \epsilon\mathbf{m_0}$ and $L_1 = \mathbf{n_1} + \epsilon\mathbf{m_1}$, where $\mathbf{m_0} = \vec{r_0}\wedge\vec{n_0}$ and $\mathbf{m_1} = \vec{r_1}\wedge\vec{n_1}$, $\vec{r_0}$ and $\vec{r_1}$ being arbitrary points belonging to lines L_0 and L_1. "Shortest" refers to the smallest rotation angle θ and translation distance d along the screw axis. Indeed, in 3D space, two lines do not usually intersect. d represents the shortest distance between the two lines, and is therefore non-zero when the lines do not intersect. We can write:

$$\begin{aligned} L_1 &= L_0(L_0^{-1}L_1) \\ &= L_0\left(\underbrace{L_0^* L_1}_{S}\right) \end{aligned}$$

since L_0 and L_1 are unitary dual quaternions. It can be shown that S indeed represents the shortest screw motion transformation between L_0 and L_1 and has the following form:

$$\begin{aligned} S = L_0^* L_1 = \cos\theta + \sin\theta\mathbf{n_S} + \epsilon\,(&-d\sin\theta + d\mathbf{n_S}\cos\theta \\ &+(\vec{a_S}\wedge\vec{n_S})\sin\theta) \end{aligned}$$

Therefore, we can compute the angle θ and distance d between any two lines in space using this expression.

C.3. Screw motion velocity between two moving lines

Following what we did in the **Appendix C.2**, we can develop the derivatives of the screw motion dual quaternion S, to show how to extract quantities like the rotation angle derivative $\dot{\theta}$ or the screw motion translation derivative, $\dot{d}$. Indeed, if we derive S, we directly obtain:

$$\begin{aligned}\dot{S} = &-\dot{\theta}\sin\theta + (\dot{\theta}\mathbf{n_S}\cos\theta + \dot{\mathbf{n}}_\mathbf{S}\sin\theta) + \epsilon\Big[-(\dot{d}\sin\theta \\ &+ d\dot{\theta}\cos\theta) + \big(\dot{d}\mathbf{n_S}\cos\theta + d\dot{\mathbf{n}}_\mathbf{S}\cos\theta - d\dot{\theta}\mathbf{n_S}\sin\theta \\ &+ (\vec{a}_S \wedge \vec{n}_S)\dot{\theta}\cos\theta + (\dot{\vec{a}}_S \wedge \vec{n}_S)\sin\theta + (\vec{a}_S \wedge \dot{\vec{n}}_S)\sin\theta\big)\Big]\end{aligned}$$

Although this expression is quite long and difficult to interpret, it is quite easy to retrieve $\dot{\theta}$ and $\dot{d}$ from this dual quaternion, provided we retrieve the parameters of the screw motion S first.

D. CONVERSION FROM AND TO FICK COORDINATES

In diverse applications, we might want to go from a rotation quaternion expression to Fick coordinates, or vice-versa. Here we briefly describe these mathematical transformations.

Fick coordinates are commonly used in the 3D eye literature (see Haslwanter, 1995, for more comprehensive explanations): they describe a 3D rotation by three successive rotations in a well-defined order. Illustrating the context of a 3D eye position, the first Fick coordinate is a horizontal rotation of θ_F around a head-fixed axis (θ_F is positive when the rotation is to the left). Then the second Fick coordinate is a vertical rotation of φ_F around the inter-aural axis (i.e., the spatial horizontal axis rotated by the first Fick rotation). φ_F is positive when the rotation is down. Finally, the third Fick coordinate is a torsional rotation of ψ_F around the line of sight (i.e., the spatial backward–forward axis rotated by the two first Fick rotations). ψ_F is positive when the rotation is clockwise. All the definitions given above are defined from the point of view of the object being rotated.

A rotation quaternion is described by a rotation angle θ and a rotation axis $\mathbf{n}$ and is written (see Equation 3):

$$R_{\text{rot}} = \cos\left(\frac{\theta}{2}\right) + \mathbf{n}\sin\left(\frac{\theta}{2}\right)$$

D.1. Fick coordinates to rotation rotor

Here, we want to compute θ and $\mathbf{n}$ from Fick coordinates (θ_F, φ_F, ψ_F). First, we compute the 3×3 rotation matrix describing the rotation (see Haslwanter, 1995), using $\mathbf{c}$ and $\mathbf{s}$ as shortcuts for cos and sin:

$$\mathbf{R}_{\text{Fick}} = \begin{pmatrix} \mathbf{c}\theta_F\mathbf{c}\varphi_F & \mathbf{c}\theta_F\mathbf{s}\varphi_F\mathbf{s}\psi_F - \mathbf{s}\theta_F\mathbf{c}\psi_F & \mathbf{c}\theta_F\mathbf{s}\varphi_F\mathbf{c}\psi_F + \mathbf{s}\theta_F\mathbf{s}\psi_F \\ \mathbf{s}\theta_F\mathbf{c}\varphi_F & \mathbf{s}\theta_F\mathbf{s}\varphi_F\mathbf{s}\psi_F + \mathbf{c}\theta_F\mathbf{c}\psi_F & \mathbf{s}\theta_F\mathbf{s}\varphi_F\mathbf{c}\psi_F - \mathbf{c}\theta_F\mathbf{s}\psi_F \\ -\mathbf{s}\varphi_F & \mathbf{c}\varphi_F\mathbf{s}\psi_F & \mathbf{c}\varphi_F\mathbf{c}\psi_F \end{pmatrix} \tag{D.1}$$

Then, we need to compute the rotor $R_{\text{rot}} = q_0 + \mathbf{q}$ from the rotation matrix. Funda et al. (1990) showed that if the rotation matrix $\mathbf{R}$ is written:

$$\mathbf{R} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

then, q_0 and $\mathbf{q} = (q_1, q_2, q_3)$ are given by:

$$\begin{aligned} q_0 &= \frac{\sqrt{1 + a_{11} + a_{22} + a_{33}}}{2} \\ q_1 &= (a_{32} - a_{23})/(4q_0) \\ q_2 &= (a_{13} - a_{31})/(4q_0) \\ q_3 &= (a_{21} - a_{12})/(4q_0) \end{aligned}$$

This transformation is for instance needed to reconstruct the 3D eye rotation quaternion, from Fick coordinates provided by a video-based eye tracking system.

D.2. Rotation rotor to Fick coordinates

Here we compute the Fick coordinates (θ_F, φ_F, ψ_F) from a rotation quaternion representation $R_{\text{rot}} = q_0 + \mathbf{q} = \cos\frac{\theta}{2} + \mathbf{n}\sin\frac{\theta}{2}$. From q_0 and $\mathbf{q} = (q_1, q_2, q_3)$, we can retrieve the rotation matrix $\mathbf{R}$ (see Funda et al., 1990):

$$\mathbf{R} = \begin{pmatrix} 1 - 2q_2^2 - 2q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1 - 2q_1^2 - 2q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1 - 2q_1^2 - 2q_2^2 \end{pmatrix} \tag{D.2}$$

Then, knowing the structure of $\mathbf{R}$ in Fick coordinates (Equation D.1), we can identify term by term and extract the Fick parameters (except when there are singularities, in which case one angle can have multiple values).

E. DERIVATION OF P^{EE} AND $\dot{P}^{\text{EE}}$

First we derive step by step the expressions for the pointing target P in eye coordinates. First we show that:

$$P^{\text{EE}} = R_{\text{EH}}T_{\text{HE}}R_{\text{HS}}T_{\text{SH}}P^{\text{SS}}T_{\text{SH}}R^*_{\text{HS}}T_{\text{HE}}R^*_{\text{EH}} \tag{E.1}$$

Indeed, we obtain this result by applying one operation (rotation or translation) at a time:

$$\begin{aligned} P^{\text{EE}} &= R_{\text{EH}}P^{\text{EH}}R^*_{\text{EH}} \\ &= R_{\text{EH}}T_{\text{HE}}P^{\text{HH}}T_{\text{HE}}R^*_{\text{EH}} \\ &= R_{\text{EH}}T_{\text{HE}}R_{\text{HS}}P^{\text{HS}}R^*_{\text{HS}}T_{\text{HE}}R^*_{\text{EH}} \\ &= R_{\text{EH}}T_{\text{HE}}R_{\text{HS}}T_{\text{SH}}P^{\text{SS}}T_{\text{SH}}R^*_{\text{HS}}T_{\text{HE}}R^*_{\text{EH}} \end{aligned}$$

where P^{EH}, P^{HH}, and P^{HS} are the coordinates of the pointing target in eye-centered head-fixed coordinates, head-centered head-fixed coordinates, and head-centered shoulder-fixed coordinates.

For the P velocity, we differentiate Equation (E.1). Differentiating the first line, we write:

$$\dot{P}^{\text{EE}} = \dot{R}_{\text{EH}}P^{\text{EH}}R^*_{\text{EH}} + R_{\text{EH}}P^{\text{EH}}\dot{R}^*_{\text{EH}} + R_{\text{EH}}\dot{P}^{\text{EH}}R^*_{\text{EH}} \tag{E.2}$$

Using the rotational velocity operator (Equation 4), Equation (E.2) may be written:

$$\begin{aligned}\dot{P^{\mathrm{EE}}} &= \frac{1}{2}R_{\mathrm{EH}}(\mathbf{\Omega_{EH}}P^{\mathrm{EH}} - P^{\mathrm{EH}}\mathbf{\Omega_{EH}})R^*_{\mathrm{EH}} + R_{\mathrm{EH}}\dot{P^{\mathrm{EH}}}R^*_{\mathrm{EH}} \\ &= R_{\mathrm{EH}}\left(\frac{1}{2}(\mathbf{\Omega_{EH}}P^{\mathrm{EH}} - P^{\mathrm{EH}}\mathbf{\Omega_{EH}}) + \dot{P^{\mathrm{EH}}}\right)R^*_{\mathrm{EH}}\end{aligned}$$

Because T_{HE} is time-invariant (the offset between eye and head rotation centers is obviously fixed in a head-fixed reference frame), we have then

$$\dot{P^{\mathrm{EH}}} = \dot{P^{\mathrm{HH}}}$$

Similarly to Equation (E.2), we have:

$$\begin{aligned}\dot{P^{\mathrm{HH}}} &= \dot{R_{\mathrm{HS}}}P^{\mathrm{HS}}R^*_{\mathrm{HS}} + R_{\mathrm{HS}}P^{\mathrm{HS}}\dot{R^*_{\mathrm{HS}}} + R_{\mathrm{HS}}\dot{P^{\mathrm{HS}}}R^*_{\mathrm{HS}} \\ &= R_{\mathrm{HS}}\left(\frac{1}{2}(\mathbf{\Omega_{HS}}P^{\mathrm{HS}} - P^{\mathrm{HS}}\mathbf{\Omega_{HS}}) + \dot{P^{\mathrm{HS}}}\right)R^*_{\mathrm{HS}}\end{aligned} \tag{E.3}$$

Finally, we assume T_{HS} is time-invariant. Thus,

$$\dot{P^{\mathrm{HS}}} = \dot{P^{\mathrm{SS}}}$$

The final expression for P velocity in an eye-centered eye-fixed reference frame is therefore:

$$\begin{aligned}\dot{P^{\mathrm{EE}}} &= R_{\mathrm{EH}}\left[\frac{1}{2}(\mathbf{\Omega_{EH}}P^{\mathrm{EH}} - P^{\mathrm{EH}}\mathbf{\Omega_{EH}})\right. \\ &\quad \left. + R_{\mathrm{HS}}\left(\frac{1}{2}(\mathbf{\Omega_{HS}}P^{\mathrm{HS}} - P^{\mathrm{HS}}\mathbf{\Omega_{HS}}) + \dot{P^{\mathrm{SS}}}\right)R^*_{\mathrm{HS}}\right]R^*_{\mathrm{EH}} \\ &= \frac{1}{2}R_{\mathrm{EH}}(\mathbf{\Omega_{EH}}P^{\mathrm{EH}} - P^{\mathrm{EH}}\mathbf{\Omega_{EH}})R^*_{\mathrm{EH}} \\ &\quad + \frac{1}{2}R_{\mathrm{EH}}R_{\mathrm{HS}}(\mathbf{\Omega_{HS}}P^{\mathrm{HS}} - P^{\mathrm{HS}}\mathbf{\Omega_{HS}})R^*_{\mathrm{HS}}R^*_{\mathrm{EH}} \\ &\quad + R_{\mathrm{EH}}R_{\mathrm{HS}}\dot{P^{\mathrm{SS}}}R^*_{\mathrm{HS}}R^*_{\mathrm{EH}}\end{aligned} \tag{E.4}$$

F. PROOF OF $\dot{D}_P$ EXPRESSION

Here we compute the expression of $\dot{\mathbf{d}}_{\mathbf{P}}$, which is the rate of change of the distance between the eye and the target P. Without loss of generality, we can write:

$$P^{\mathrm{EE}} = 1 + \epsilon\begin{pmatrix}x(t) & y(t) & z(t)\end{pmatrix}$$

Therefore,

$$\dot{P^{\mathrm{EE}}} = 0 + \epsilon\begin{pmatrix}\dot{x}(t) & \dot{y}(t) & \dot{z}(t)\end{pmatrix}$$

The distance between the eye and the target P, denoted d_P, is computed from the expression of P^{EE}:

$$d_P = \sqrt{x^2 + y^2 + z^2}$$

Then, $\dot{d_P}$ can be developed:

$$\begin{aligned}\dot{d_P} &= \frac{x\dot{x} + y\dot{y} + z\dot{z}}{\sqrt{x^2 + y^2 + z^2}} \\ &= \frac{\mathbf{P^{EE}}\cdot\dot{\mathbf{P}}_{\mathbf{EE}}}{d_P}\end{aligned}$$

G. FORWARD AND INVERSE KINEMATICS: DERIVATION OF RESULTS

G.1. Computation of the 2D velocity of the end-effector position

We show that:

$$\dot{P^{\mathrm{BS}}} = \underbrace{\left(\frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{n} - \mathbf{n}P_{\mathrm{BS}}\right) \quad R^*_{\mathrm{UB}}\frac{P_{\mathrm{UE}}\mathbf{n} - \mathbf{n}P_{\mathrm{UE}}}{2}R_{\mathrm{UB}}\right)}_{J_{\mathrm{DQ}}}\begin{pmatrix}\dot{\varphi} & \dot{\theta}\end{pmatrix}^T$$

Indeed,

$$\begin{aligned}\dot{P^{\mathrm{BS}}} &= \frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{n}\dot{\varphi} - \dot{\varphi}\mathbf{n}P_{\mathrm{BS}}\right) + R^*_{\mathrm{UB}}\frac{P_{\mathrm{UE}}\mathbf{n}\dot{\theta} - \dot{\theta}\mathbf{n}P_{\mathrm{UE}}}{2}R_{\mathrm{UB}} \\ &= \left[\frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{n} - \mathbf{n}P_{\mathrm{BS}}\right)\right]\dot{\varphi} + \left[R^*_{\mathrm{UB}}\frac{P_{\mathrm{UE}}\mathbf{n} - \mathbf{n}P_{\mathrm{UE}}}{2}R_{\mathrm{UB}}\right]\dot{\theta} \\ &= \underbrace{\left(\frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{n} - \mathbf{n}P_{\mathrm{BS}}\right) \quad R^*_{\mathrm{UB}}\frac{P_{\mathrm{UE}}\mathbf{n} - \mathbf{n}P_{\mathrm{UE}}}{2}R_{\mathrm{UB}}\right)}_{J_{\mathrm{DQ}}}\begin{pmatrix}\dot{\varphi} & \dot{\theta}\end{pmatrix}^T\end{aligned}$$

where $\mathbf{n}$ is the rotation axis of the shoulder and the elbow (since we only consider planar motions)

G.2. Proof that $\frac{1}{2}\left(P_{BS}\Omega_{UB} - \Omega_{UB}P_{BS}\right) = -\epsilon\left(\vec{P}_{BS}\wedge\vec{\Omega}_{UB}\right)$

First, using the definition of a point position dual quaternion $P_{\mathrm{BS}} = 1 + \epsilon\mathbf{P_{BS}}$, we develop the left side of this equation:

$$\begin{aligned}\frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}}P_{\mathrm{BS}}\right) &= \frac{1}{2}\left[(1 + \epsilon\mathbf{P_{BS}})\mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}}(1 + \epsilon\mathbf{P_{BS}})\right] \\ &= \epsilon\frac{1}{2}\left(\mathbf{P_{BS}}\mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}}\mathbf{P_{BS}}\right)\end{aligned}$$

Then, let us compute the quaternion multiplications of the two bivectors $\mathbf{P_{BS}}$ and $\mathbf{\Omega_{UB}}$, using the quaternion properties developed in section 2.1:

$$\begin{aligned}\mathbf{P_{BS}}\mathbf{\Omega_{UB}} &= \left(i\vec{P}_{\mathrm{BS}}\right)\left(i\vec{\Omega}_{\mathrm{UB}}\right) \\ &= -\vec{P}_{\mathrm{BS}}\vec{\Omega}_{\mathrm{UB}} \\ &= -\vec{P}_{\mathrm{BS}}\cdot\vec{\Omega}_{\mathrm{UB}} - \vec{P}_{\mathrm{BS}}\wedge\vec{\Omega}_{\mathrm{UB}}\end{aligned}$$

Similarly, we have:

$$\mathbf{\Omega_{UB}}\mathbf{P_{BS}} = -\vec{\Omega}_{\mathrm{UB}}\cdot\vec{P}_{\mathrm{BS}} - \vec{\Omega}_{\mathrm{UB}}\wedge\vec{P}_{\mathrm{BS}}$$

Therefore,

$$\begin{aligned}\frac{1}{2}\left(P_{\mathrm{BS}}\mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}}P_{\mathrm{BS}}\right) &= \epsilon\frac{1}{2}\left(\mathbf{P_{BS}}\mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}}\mathbf{P_{BS}}\right) \\ &= \epsilon\frac{1}{2}\left(-\vec{P}_{\mathrm{BS}}\wedge\vec{\Omega}_{\mathrm{UB}} - (-\vec{\Omega}_{\mathrm{UB}}\wedge\vec{P}_{\mathrm{BS}})\right)\end{aligned}$$

$$= \epsilon \frac{1}{2} \left(-\vec{P}_{\text{BS}} \wedge \vec{\Omega}_{\text{UB}} - (\vec{P}_{\text{BS}} \wedge \vec{\Omega}_{\text{UB}}) \right)$$
$$= -\epsilon \left(\vec{P}_{\text{BS}} \wedge \vec{\Omega}_{\text{UB}} \right)$$

G.3. Derivation of the matrix expression for $\dot{P}_{BS}$

Here, we start from Equation (19) which expresses the velocity of the point end-effector P as a function of the rotational velocities of the shoulder and elbow, using the dual quaternion formalism. As a reminder, this equation is:

$$\dot{P}_{\text{BS}} = \frac{1}{2} \left(P_{\text{BS}} \mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}} P_{\text{BS}} \right) + R^*_{\text{UB}} \frac{P_{\text{UE}} \mathbf{\Omega_{LU}} - \mathbf{\Omega_{LU}} P_{\text{UE}}}{2} R_{\text{UB}} \quad \text{(G.1)}$$

where P_{UE} is the position dual quaternion of point P in an **u**pper-arm fixed **e**lbow-centered reference frame:

$$P_{\text{UE}} = R^*_{\text{LU}} P_{\text{LE}} R_{\text{LU}}$$

Using the relationship derived in **Appendix G.2**,

$$\frac{1}{2} \left(P_{\text{BS}} \mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}} P_{\text{BS}} \right) = -\epsilon \left(\vec{P}_{\text{BS}} \wedge \vec{\Omega}_{\text{UB}} \right)$$

we can develop Equation (G.1) as:

$$\begin{aligned} \dot{P}_{\text{BS}} &= \frac{1}{2} \left(P_{\text{BS}} \mathbf{\Omega_{UB}} - \mathbf{\Omega_{UB}} P_{\text{BS}} \right) + R^*_{\text{UB}} \frac{P_{\text{UE}} \mathbf{\Omega_{LU}} - \mathbf{\Omega_{LU}} P_{\text{UE}}}{2} R_{\text{UB}} \\ &= -\epsilon \left(\vec{P}_{\text{BS}} \wedge \vec{\Omega}_{\text{UB}} \right) + R^*_{\text{UB}} \left[-\epsilon \left(\vec{P}_{\text{UE}} \wedge \vec{\Omega}_{\text{LU}} \right) \right] R_{\text{UB}} \\ &= -\epsilon \left[\left(\vec{P}_{\text{BS}} \wedge \vec{\Omega}_{\text{UB}} \right) + R^*_{\text{UB}} \left(\vec{P}_{\text{UE}} \wedge \vec{\Omega}_{\text{LU}} \right) R_{\text{UB}} \right] \\ &= -\epsilon \left[i \left(\vec{P}_{\text{BS}} \times \vec{\Omega}_{\text{UB}} \right) + R^*_{\text{UB}} i \left(\vec{P}_{\text{UE}} \times \vec{\Omega}_{\text{LU}} \right) R_{\text{UB}} \right] \end{aligned}$$

Using the rotation matrix R^M corresponding to the rotation dual quaternion R [see Equation (D.2) in **Appendix D** for how to compute this rotation matrix from the rotation dual quaternion], the rotation of a point P, written as $P' = R^* P R$ in dual quaternion formalism, is written as $\vec{P}' = R^M \vec{P}$ in matrix notation. Furthermore, a cross product of two vectors $\vec{v} = \vec{a} \times \vec{b}$ can also be written as $\vec{v} = \tilde{A}_{\mathbf{a}} \vec{b}$ where

$$\tilde{A}_{\mathbf{a}} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

where $\mathbf{a} = (a_1\ a_2\ a_3)$. Note that this matrix has always rank 2 since for an equation in $\vec{x}$: $\vec{a} \times \vec{x} = \vec{v}$, the component of $\vec{x}$ parallel to $\vec{a}$ can be arbitrary. All the solutions $\vec{x}$ lie on a line.

Using these two properties, $\dot{P}_{\text{BS}}$ can be developed further and its vector velocity component, $\dot{\mathbf{P}}_{\mathbf{BS}}$, can be written in matrix form as:

$$\dot{\mathbf{P}}_{\mathbf{BS}} = -\begin{pmatrix} \tilde{A}_{\mathbf{P_{BS}}} & R^M_{\text{UB}} \tilde{A}_{\mathbf{P_{UE}}} \end{pmatrix} \begin{pmatrix} \mathbf{\Omega_{UB}} \\ \mathbf{\Omega_{LU}} \end{pmatrix}$$

G.4. Computation of the pseudo-inverse J^+

In this appendix, we provide a brief tutorial on the computation of the pseudo-inverse J^+ to the interested reader. We also explain how to deal with solutions which are close to singularities.

In practice, Equation (24) requires that we compute explicitly the pseudo-inverse J^+, by applying for example an algorithm based on the svd (singular value decomposition) of J [see Klein and Huang (1983), it is also what Matlab does to compute it]. The advantage is that the pseudo-inverse exists whatever the rank k of the matrix J which is of size $n \times m$, even if $k < \min(n, m)$. However, the computation time is quite large. If the matrix J has rank $k = \min(n, m)$, we can solve $J\mathbf{x} = \dot{\mathbf{v}}$ (where $\dot{\mathbf{v}}$ is the velocity of the end-effector) without explicitly computing the pseudo-inverse (Klein and Huang, 1983). Indeed, in the case where the system is underdetermined, $k = m, m < n$, the pseudo-inverse is:

$$J^+ = J^T (JJ^T)^{-1}$$

while if the system is overdetermined, $k = n, n < m$, then:

$$J^+ = (J^T J)^{-1} J^T$$

Taking advantage of this formulation, if the system is underdetermined, we solve classically for:

$$(JJ^T)\mathbf{w} = \dot{\mathbf{v}} \quad \text{(G.2)}$$

where $\mathbf{w}$ is an intermediate variable from which $\mathbf{x}$ is computed by taking$\mathbf{x} = J^T \mathbf{w}$. If the system is overdetermined, then the solution $\mathbf{x}$ is obtained by solving:

$$(J^T J)\mathbf{x} = J^T \dot{\mathbf{v}} \quad \text{(G.3)}$$

Using these methods, a solution for $\mathbf{x}$ is obtained much faster than by explicitly computing the pseudo-inverse J^+. However, these methods do not work properly if there are singularities in the kinematic system. Singularities happen for example when the two links of the arm are aligned. In this case, the rank of the jacobian J is diminished by at least 1, and we can no longer apply this technique. Either, we manually remove one linearly dependent row of J, or we compute the explicit pseudo-inverse. Moreover, it can be that there are no solutions when a singularity is met. If the two links are aligned and if the velocity of the end-effector has a component aligned with the two links, there are no solutions (see Sciavicco and Siciliano, 1996).

Finally, in the vicinity of a singularity, the joint velocities may become very large. A solution to this last problem can be found by applying the *damped least-squares (DLS) inverse* instead of the pseudo-inverse. It consists in using the term $JJ^T + k_0 I$ (where I is the corresponding identity matrix, and k_0 is a scalar to be chosen) instead of JJ^T in Equation (G.2) or $J^T J + k_0 I$ in Equation (G.3). The equations are therefore better conditioned. As k_0 increases, the joint velocities decrease but at the price that they do not obey perfectly the equations $J\mathbf{x} = \dot{\mathbf{v}}$.

G.5. n-link arm: forward and inverse kinematics of the end-effector position

In the main text, we have introduced the problem of inverse kinematics and its various difficulties through the example of the four-dof two-link arm: how to deal with redundancy, how to deal with singularities, how to deal with imperfect numerical integration. In the following, we describe how to build the jacobian matrix for a general serial manipulator (something like an arm with n links). The methodology and the problems are the same but the size of the system is bigger.

The forward kinematic of a n-link arm end-effector position is:

$$\begin{aligned} P^N &= \left(R_N^* T_{N-1} R_{N-1}^* \dots T_1 R_1^*\right) P^0 \left(R_1 T_1 \dots R_{N_1} T_{N-1} R_N\right) \\ &= \underbrace{R_N^* \left(\prod_{i=1}^{N-1} T_i R_i^*\right)}_{\tilde{S}_{\text{tot}}^*} P^0 \underbrace{\left(\prod_{i=1}^{N-1} R_i T_i\right) R_N}_{S_{\text{tot}}} \end{aligned}$$

where P^0 is the end-effector position dual quaternion in the end-effector fixed reference frame, $R_i = \cos\frac{\theta_i}{2} + \mathbf{n}_i \sin\frac{\theta_i}{2}$ is the rotation quaternion representing the rotation from link $N-i+1$ to link $N-i$ (link 0 is the inertial base reference frame) in $N-i$ link-fixed coordinates, and $T_i = 1 + \epsilon\frac{d_i}{2}\mathbf{t}_i$ is the translation from link $N-i+1$ to link $N-i$ (link 0 is the inertial base reference frame) in $N-i$ link-fixed coordinates.

Its velocity can be derived:

$$\begin{aligned} \dot{P^N} = &\left(\prod_{j=1}^{N} R_j\right)^* \dot{P^0} \left(\prod_{j=1}^{N} R_j\right) \\ &+ \sum_{i=1}^{N-1} \left[\left(\prod_{j=i+1}^{N} R_j\right)^* 2\dot{T}_i \left(\prod_{j=i+1}^{N} R_j\right)\right] \\ &+ \sum_{i=1}^{N} \left[\left(\prod_{j=i+1}^{N+1} R_j\right)^* \frac{P^i \mathbf{\Omega}_i - \mathbf{\Omega}_i P^i}{2} \left(\prod_{j=i+1}^{N+1} R_j\right)\right] \end{aligned}$$

where $R_{N+1} = 1$, which is set in order to avoid writing one additional line for the term of the sum where $i = N$, and where

$$P^i = \left[\left(\prod_{j=1}^{i-1} R_j T_j\right) R_i\right]^* P^0 \left[\left(\prod_{j=1}^{i-1} R_j T_j\right) R_i\right]$$

where $\dot{T}_i$ is the translation velocity dual quaternion from link $N-i+1$ to link $N-i$, $\mathbf{\Omega}_i$ is the rotational velocity between link $N-i+1$ to link $N-i$, and $\dot{P^0}$ is the end-effector velocity dual quaternion in the end-effector fixed reference frame (most of the time, $\dot{P^0} = 0$). From this expression, we can build the jacobian J and then write the inverse kinematic equations $J\mathbf{x} = \dot{\mathbf{v}}$:

$$\left(\underbrace{\prod_{j=1}^{N} R_{N-j+1}^M \cdots \prod_{j=1}^{N-i} R_{N-j+1}^M \cdots\cdots \left(\prod_{j=0}^{N-i} R_{N-j+1}^M\right)\left(-\tilde{A}_{Pi}\right) \cdots}_{i=1,\dots,N-1 \qquad i=1,\dots,N}\right) \begin{pmatrix} \dot{P}^0 \\ \vdots \\ \dot{T}_i \\ \vdots \\ \Omega_i \\ \vdots \end{pmatrix} = \dot{\mathbf{P}}^N$$

Then, we predict the rotation and translation dual quaternions at time $t + \Delta t$, $R_j(t + \Delta t)$, and $T_j(t + \Delta t)$, similarly to Equation (25).

G.6. Threshold value for the feedback term in the numerical procedure

As explained in section 3.2.3 where we developed the numerical simulation example, we can take into account the difference between the true end-effector position and the end-effector position estimated through the reconstructed joint angles. To do so, we tune a parameter, k_1. The larger k_1 the smaller is the position error, because we penalize the position error more with larger values of k_1. But k_1 can not be too large for discrete time systems. If k_1 is too large, the system will be unstable and the error will grow. The *threshold* value for k_1 depends on the simulation step Δt. Indeed, the position error is described by a linear differential equation (see Sciavicco and Siciliano, 1996):

$$\dot{\mathbf{e}} + K\mathbf{e} = \mathbf{0}$$

where $\mathbf{e}$ is the position error and K is the *feedback* matrix, that we fix for the following to be equal to k_1 multiplied by the identity matrix I (we penalize the errors in all the dimensions similarly). In discrete time, we have:

$$\frac{\mathbf{e}[k + \Delta t] - \mathbf{e}[k]}{\Delta t} = -K\mathbf{e}[k]$$

which is reformulated as:

$$\begin{aligned} \mathbf{e}[k + \Delta t] &= (I - K\Delta t)\mathbf{e}[k] \\ &= (1 - k_1 \Delta t) I \mathbf{e}[k] \end{aligned}$$

From this expression, we see that $0 < k_1 < \frac{2}{\Delta t}$ for the system to be stable (in discrete time, the eigenvalues must lie inside the unit imaginary disk in order to be stable).

H. INVERSE KINEMATICS FROM THE END-EFFECTOR POSITION AND ORIENTATION

H.1. Three-link arm

In order to build the jacobian matrix, we need to transform Equation (28) [where $\dot{S}_x$ is the unknown and $\dot{L}(t)$ is known for our inverse kinematics application] into a matrix equation.

First, we need to use the matrix expression for a screw motion transformation applied to a line or a line velocity (which have the same structure). Let us consider a line $L_0 = \mathbf{n}_0 + \epsilon\mathbf{m}_0$ to which we apply a general screw motion S: a rotation of angle θ around a line whose orientation is $\mathbf{s}$ and that is offset by $\mathbf{a}$ from the origin

(it can be the position of any point on the line), followed by a translation d along the line axis $\mathbf{s}$. Then, the expression:

$$L(t) = S^* L_0 S = \mathbf{n}(t) + \epsilon \mathbf{m}(t) \tag{H.1}$$

can be described by the following matrix expression (see proof in **Appendix I.1**):

$$\underbrace{\begin{pmatrix} R^M & \mathbf{0}_{3\times3} \\ \tilde{A}_{\mathbf{a}} R^M - R^M \tilde{A}_{\mathbf{a}} + R^M d \tilde{A}_{\mathbf{s}} & R^M \end{pmatrix}}_{J} \begin{pmatrix} \mathbf{n}_0 \\ \mathbf{m}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{n} \\ \mathbf{m} \end{pmatrix} \tag{H.2}$$

where R^M is the rotation matrix associated with the rotation quaternion $\cos(\theta/2) + \mathbf{s}\sin(\theta/2)$, $\mathbf{0}$ is the 3×3 matrix whose entries are all 0 and $\tilde{A}_{\mathbf{a}}$ (resp. $\tilde{A}_{\mathbf{s}}$) is the anti-symmetric matrix associated with the cross product, $\mathbf{a} \times \cdot$ (resp. $\mathbf{s} \times \cdot$).

If we transform L_0 through N screw motions $S_1, \ldots, S_N$:

$$L(t) = \left(\prod_{i=1}^{N} S_i\right)^* L_0 \left(\prod_{i=1}^{N} S_i\right) = \mathbf{n}(t) + \epsilon \mathbf{m}(t) \tag{H.3}$$

then the corresponding matrix expression can be computed by combining N matrices of the type of J in Equation (H.2):

$$\prod_{i=1}^{N} J_{N-i+1} \begin{pmatrix} \mathbf{n}_0 \\ \mathbf{m}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{n} \\ \mathbf{m} \end{pmatrix} \tag{H.4}$$

Let us consider the expression of $L(t)$ given by Equation (H.1). The line velocity, $\dot{L}(t)$ is given by:

$$\begin{aligned} \dot{L}(t) &= \dot{S}^* L_0 S + S^* L_0 \dot{S} + S^* \dot{L_0} S \\ &= \dot{\mathbf{n}}(t) + \epsilon \dot{\mathbf{m}}(t) \end{aligned} \tag{H.5}$$

where $\dot{L_0} = \dot{\mathbf{n}}_0 + \epsilon \dot{\mathbf{m}}_0$, S is the screw motion operator and $\dot{S}$ its velocity. Let us write the matrix expression of this equation. The third term of Equation (H.5) consists simply of a screw motion transformation of the same form as shown in Equation (H.1), applied to the line velocity. The sum of the two first terms gathers the velocity component due to the screw motion velocity applied to the line L_0. In **Appendix I.2**, we show that if we parameterize S as $S = TT_L^* R T_L$ as described in sections 2.5.1 and 2.5.2, then we have:

$$\begin{aligned} \dot{S}^* L_0 S + S^* L_0 \dot{S} = &-R^* \left(\epsilon \dot{d} \left(\vec{n}_0 \wedge \vec{s}\right)\right) R \\ &+ R^* \left(\epsilon \left(\vec{n}_0 \wedge \dot{\vec{a}}\right)\right) R - \epsilon \left(\left(R^* \mathbf{n}_0 R\right) \wedge \dot{\vec{a}}\right) \\ &+ T_L^* \left(\vec{\Omega} \wedge \vec{n}_1 + \epsilon \left(\vec{\Omega} \wedge \vec{m}_1\right)\right) T_L \end{aligned} \tag{H.6}$$

where $\mathbf{n}_1 + \epsilon \mathbf{m}_1 = \left(TT_L^* R\right)^* L_0 \left(TT_L^* R\right)$. The first term of Equation (H.6) represents the influence of the translation velocity $\dot{d}$ along the screw axis $\mathbf{s}$ onto the $\dot{\mathbf{m}}$ component of $\dot{L}$. The second term represents the influence of the offset velocity $\dot{\mathbf{a}}$ between the screw motion line and the origin onto the $\dot{\mathbf{m}}$ component again. Finally, the last term represents the influence of the rotational velocity $\boldsymbol{\Omega}$ around the screw motion line onto both the $\dot{\mathbf{n}}$ and $\dot{\mathbf{m}}$ components of the line velocity. In matrix terms, Equation (H.6) may be written:

$$\underbrace{\begin{pmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & -\tilde{A}_{\mathbf{n_1}} \\ -R^M \left(\vec{n}_0 \wedge \vec{s}\right) & R^M \tilde{A}_{\mathbf{n}_0} - \tilde{A}_{R^M \mathbf{n}_0} & -\tilde{A}_{\mathbf{a}} \tilde{A}_{\mathbf{n_1}} - \tilde{A}_{\mathbf{m_1}} \end{pmatrix}}_{K} \begin{pmatrix} \dot{d} \\ \dot{\vec{a}} \\ \vec{\Omega} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{n}} \\ \dot{\mathbf{m}} \end{pmatrix} \tag{H.7}$$

From all these expressions, we can now build the matrix expression for Equation (28), knowing that for our application, $\dot{L_0} = 0$ (the screwdriver end-effector has no intrinsic velocity in the reference configuration). Therefore we have:

$$\underbrace{\begin{pmatrix} J_{\mathrm{UB}} J_{\mathrm{LU}} K_{\mathrm{HL}} & J_{\mathrm{UB}} K_{\mathrm{LU}} & K_{\mathrm{UB}} \\ \mathbf{0}_{3\times3} & \tilde{A}_{\mathbf{n}_{\mathrm{LU}}} & \mathbf{0}_{3\times3} \end{pmatrix}}_{A} \begin{pmatrix} \boldsymbol{\Omega}_{\mathbf{HL}} \\ \boldsymbol{\Omega}_{\mathbf{LU}} \\ \boldsymbol{\Omega}_{\mathbf{UB}} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{n}} \\ \dot{\mathbf{m}} \\ \mathbf{0}_{3\times1} \end{pmatrix} \tag{H.8}$$

where A is the jacobian matrix of size 9×9 of rank 8, which is consistent with the fact that there are seven-dof and six parameters coding for the line L. In Equation (H.8), we have:

- J_{UB}, J_{LU}, and J_{HL} are matrices of the form J described in Equation (H.2). J_{UB} has only a rotational component, while J_{LU} and J_{HL} also have an offset component for their rotation axis, $\mathbf{a_{LU}}$ and $\mathbf{a_{HL}}$.
- K_{UB}, K_{LU}, and K_{HL} are matrices of the form of the last column of matrix K in Equation (H.7), since in the seven-dof three-link arm, we only consider rotational velocities. They are therefore of the type: $K_i = \begin{pmatrix} -\tilde{A}_{\mathbf{n}'_\mathbf{i}} \\ -\tilde{A}_{\mathbf{a}_i} \tilde{A}_{\mathbf{n}'_\mathbf{i}} - \tilde{A}_{\mathbf{m}'_\mathbf{i}} \end{pmatrix}$ for $i = 1, 2, 3$ where $\mathbf{n}'_\mathbf{i}$ and $\mathbf{m}'_\mathbf{i}$ are obtained as follows:

$$\begin{pmatrix} \mathbf{n}'_\mathbf{1} \\ \mathbf{m}'_\mathbf{1} \end{pmatrix} = \begin{pmatrix} R^M_{\mathrm{UB}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & R^M_{\mathrm{UB}} \end{pmatrix} J_{\mathrm{LU}} J_{\mathrm{HL}} \begin{pmatrix} \mathbf{n_0} \\ \mathbf{m_0} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{n}'_\mathbf{2} \\ \mathbf{m}'_\mathbf{2} \end{pmatrix} = \begin{pmatrix} R^M_{\mathrm{LU}} & \mathbf{0}_{3\times3} \\ -R^M_{\mathrm{LU}} \tilde{A}_{\mathbf{a}_{\mathrm{LU}}} & R^M_{\mathrm{LU}} \end{pmatrix} J_{\mathrm{HL}} \begin{pmatrix} \mathbf{n_0} \\ \mathbf{m_0} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{n}'_\mathbf{3} \\ \mathbf{m}'_\mathbf{3} \end{pmatrix} = \begin{pmatrix} R^M_{\mathrm{HL}} & \mathbf{0}_{3\times3} \\ -R^M_{\mathrm{HL}} \tilde{A}_{\mathbf{a}_{\mathrm{HL}}} & R^M_{\mathrm{HL}} \end{pmatrix} \begin{pmatrix} \mathbf{n_0} \\ \mathbf{m_0} \end{pmatrix}$$

Then we can use the techniques described in the previous sections, using the jacobian matrix to solve the inverse kinematics problem. In the following, we generalize this method to a $n-$link arm whose end-effector orientation matters.

H.2. n-link arm

Again the end-effector reference orientation and position in the base frame is modeled by a line $L_0 = \mathbf{n}_0 + \epsilon \mathbf{m}_0$. The N-link motions can each be described by a screw motion S_i, $i = 1, \ldots, N$. Therefore we have:

$$L(t) = \left(\prod_{i=1}^{N} S_i\right)^* L_0 \left(\prod_{i=1}^{N} S_i\right) = \mathbf{n}(t) + \epsilon \mathbf{m}(t)$$

The line velocity can be expressed as:

$$\dot{L}(t) = \left(\prod_{i=1}^{N} S_i\right)^* \dot{L}_0 \left(\prod_{i=1}^{N} S_i\right) + \sum_{i=1}^{N} \left[\left(\prod_{j=i+1}^{N} S_j\right)^* \left(\dot{S}_i^* L_{i-1} S_i + S_i^* L_{i-1} \dot{L}_i\right) \left(\prod_{j=i+1}^{N} S_j\right) \right]$$

where $L_i = \left(\prod_{j=1}^{i-1} S_j\right)^* L_0 \left(\prod_{j=1}^{i-1} S_j\right)$. Now, in matrix terms, using the properties that we derived above, we obtain:

$$\underbrace{\begin{pmatrix} \prod_{i=1}^{N} J_{N-i+1} & \cdots & \overbrace{\left(\prod_{j=1}^{N-i+1} J_{N-j+1}\right) K_i}^{i=1,\ldots,N} & \cdots \end{pmatrix}}_{A} \begin{pmatrix} \dot{\mathbf{n}}_0 \\ \dot{\mathbf{m}}_0 \\ \vdots \\ \dot{d}_i \\ \dot{\mathbf{a}}_i \\ \boldsymbol{\Omega}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{n}} \\ \dot{\mathbf{m}} \end{pmatrix} \tag{H.9}$$

where J_i is of the form described by Equation (H.2), using the rotational and translational parameters of link i:

$$J_i = \begin{pmatrix} R_i^M & \mathbf{0}_{3\times3} \\ \tilde{A}_{\mathbf{a_i}} R_i^M - R_i^M \tilde{A}_{\mathbf{a_i}} + R_i^M d_i \tilde{A}_{\mathbf{s_i}} & R_i^M \end{pmatrix}$$

and where K_i has the following structure:

$$K_i = \begin{pmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & -\tilde{A}_{\mathbf{n_i'}} \\ -R_i^M(\mathbf{n_{i-1}} \times \mathbf{s_i}) & R_i^M \tilde{A}_{\mathbf{n_{i-1}}} - \tilde{A}_{R_i^M \mathbf{n_{i-1}}} & -\tilde{A}_{\mathbf{a_i}} \tilde{A}_{\mathbf{n_i'}} - \tilde{A}_{\mathbf{m_i'}} \end{pmatrix}$$

where:

- $\begin{pmatrix} \mathbf{n_i} \\ \mathbf{m_i} \end{pmatrix} = \left(\prod_{j=1}^{i} J_j\right) \begin{pmatrix} \mathbf{n_0} \\ \mathbf{m_0} \end{pmatrix}$
- $\begin{pmatrix} \mathbf{n_i'} \\ \mathbf{m_i'} \end{pmatrix} = \begin{pmatrix} R_i^M & \mathbf{0}_{3\times3} \\ -R_i^M \tilde{A}_{\mathbf{a_i}} & R_i^M \end{pmatrix} \left(\prod_{j=1}^{i-1} J_j\right) \begin{pmatrix} \mathbf{n_0} \\ \mathbf{m_0} \end{pmatrix}$

We can of course add constraints by including them into the jacobian matrix A of Equation (H.9), for example if we want to impose a rotation axis for $\boldsymbol{\Omega_i}$. Then, using this jacobian matrix A, we can proceed like described before to find a solution for the inverse kinematics problem.

I. MATRIX EXPRESSIONS FOR SCREW MOTION POSITION AND VELOCITY TRANSFORMATIONS

I.1. Screw motion transformation

Here we want to express the following line transformation under a matrix form

$$L(t) = S^* L_0 S = \mathbf{n}(t) + \epsilon \mathbf{m}(t) \tag{I.1}$$

where $L_0 = \mathbf{n_0} + \epsilon \mathbf{m_0}$. A screw motion S can be parameterized as $S = TT_L^* R T_L$ (see section 2.5.1), where R is a pure rotation dual quaternion, $T_L = 1 + \epsilon \frac{1}{2}\mathbf{a}$ represents the offset of the rotation axis from the origin and $T = 1 + \epsilon d\mathbf{s}$ is the translation along the screw axis. Therefore, we write:

$$L(t) = (T_L^* R^* T_L T^*)(\mathbf{n_0} + \epsilon \mathbf{m_0})(T T_L^* R T_L) \tag{I.2}$$

We can show that $T^*(\mathbf{n_0} + \epsilon \mathbf{m_0}) T = \mathbf{n_0} + \epsilon(\mathbf{m_0} + d\mathbf{s} \times \mathbf{n}_0)$. Performing one transformation, rotation or translation, at a time, we can write in matrix form:

$$\begin{pmatrix} \mathbf{n}(t) \\ \mathbf{m}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \tilde{A}_{\mathbf{a}} & \mathbf{I}_{3\times3} \end{pmatrix} \begin{pmatrix} R^M & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & R^M \end{pmatrix} \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ -\tilde{A}_{\mathbf{a}} & \mathbf{I}_{3\times3} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ d\tilde{A}_{\mathbf{s}} & \mathbf{I}_{3\times3} \end{pmatrix} \begin{pmatrix} \mathbf{n}_0 \\ \mathbf{m}_0 \end{pmatrix} = \underbrace{\begin{pmatrix} R^M & \mathbf{0}_{3\times3} \\ \tilde{A}_{\mathbf{a}} R^M - R^M \tilde{A}_{\mathbf{a}} + R^M d\tilde{A}_{\mathbf{s}} & R^M \end{pmatrix}}_{J} \begin{pmatrix} \mathbf{n_0} \\ \mathbf{m_0} \end{pmatrix}$$

I.2. Screw motion velocity

Here we demonstrate that if we parameterize S as $S = TT_L^* R T_L$ as described in sections 2.5.1 and 2.5.2, then we have:

$$\begin{aligned} \dot{S}^* L_0 S + S^* L_0 \dot{S} = &-R^* \left(\epsilon \dot{d} (\vec{n}_0 \wedge \vec{s})\right) R \\ &+R^* \left(\epsilon \left(\vec{n}_0 \wedge \dot{\vec{a}}\right)\right) R - \epsilon \left(\left(R^* \mathbf{n}_0 R\right) \wedge \dot{\vec{a}}\right) \\ &+T_L^* \left(\vec{\Omega} \wedge \vec{n}_1 + \epsilon \left(\vec{\Omega} \wedge \vec{m}_1\right)\right) T_L \end{aligned}$$

where $\mathbf{n_1} + \epsilon \mathbf{m_1} = \left(TT_L^* R\right)^* L_0 \left(TT_L^* R\right)$. Indeed, using Equation (13), we have:

$$\dot{S}^* L_0 S + S^* L_0 \dot{S} = \underbrace{(R^* \dot{T}^*) L_0 (TT_L^* R T_L) + (T_L^* R^* T_L T^*) L_0 (\dot{T} R)}_{\text{translational velocity component}} + \underbrace{(R^* \dot{T}_L + \dot{T}_L^* R^*) L_0 (TT_L^* R T_L) + (T_L^* R^* T_L T^*) L_0 (\dot{T}_L^* R + R \dot{T}_L)}_{\text{offset velocity component}} + \underbrace{\left(-\frac{1}{2} T_L^* \boldsymbol{\Omega} R^* T_L T^*\right) L_0 (TT_L^* R T_L) + (T_L^* R^* T_L T^*) L_0 \left(\frac{1}{2} TT_L^* R \boldsymbol{\Omega} T_L\right)}_{\text{rotational velocity component}}$$

Using the following properties:

- velocity dual quaternion due to translational velocity: $\underbrace{(0 + \epsilon \mathbf{v})}_{\dot{T}} (\mathbf{n} + \epsilon \mathbf{m}) = \epsilon \underbrace{\mathbf{vn}}_{A}$
- invariance of this velocity dual quaternion due to translations: $\underbrace{(1 + \epsilon \mathbf{a})}_{T} \epsilon A = \epsilon A$
- rotation of this velocity dual quaternion due to translations: $R^*(0 + \epsilon A) R = \epsilon (R^* A R)$

The velocity component due to the translational velocity along the screw axis can be developed as follows:

translational velocity component

$$
\begin{aligned}
&= (R^*\dot{T}^*)L_0(TT_L^*RT_L) + (T_L^*R^*T_LT^*)L_0(\dot{T}R)\\
&= R^*\dot{T}^*L_0R + (T_L^*R^*T_LT^*)L_0(\dot{T}R)\\
&= R^*\left[-\epsilon\frac{\dot{d}}{2}\mathbf{s}(\mathbf{n}_0+\epsilon\mathbf{m}_0) + (\mathbf{n}_0+\epsilon\mathbf{m}_0)\epsilon\frac{\dot{d}}{2}\mathbf{s}\right]R\\
&= R^*\left[\epsilon(-\frac{\dot{d}}{2}\mathbf{s}\mathbf{n}_0 + \frac{\dot{d}}{2}\mathbf{n}_0\mathbf{s}\right]R\\
&= -R^*\left(\epsilon\dot{d}\,(\vec{n}_0\wedge\vec{s})\right)R
\end{aligned}
$$

The velocity component due to the screw axis offset velocity can be expressed as:

offset velocity component

$$
\begin{aligned}
&= (R^*\dot{T}_L + \dot{T}_L^*R^*)L_0(TT_L^*RT_L) + (T_L^*R^*T_LT^*)L_0(\dot{T}_L^*R + R\dot{T}_L)\\
&= R^*\dot{T}_LL_0R + \dot{T}_L^*R^*L_0R + R^*L_0\dot{T}_L^*R + R^*L_0R\dot{T}_L\\
&= R^*(\dot{T}_LL_0 + L_0\dot{T}_L^*)R + \dot{T}_L^*(R^*L_0R) + (R^*L_0R)\dot{T}_L\\
&= R^*\left(\epsilon\left(\vec{n}_0\wedge\dot{\vec{a}}\right)\right)R - \epsilon\left(\left(R^*\mathbf{n}_0R\right)\wedge\dot{\vec{a}}\right)
\end{aligned}
$$

Finally, the velocity component due to the rotational velocity is:

rotational velocity component

$$
\begin{aligned}
&= \left(-\frac{1}{2}T_L^*\boldsymbol{\Omega}R^*T_LT^*\right)L_0(TT_L^*RT_L)\\
&\quad + (T_L^*R^*T_LT^*)L_0\left(\frac{1}{2}TT_L^*R\boldsymbol{\Omega}T_L\right)\\
&= -\frac{1}{2}T_L^*\boldsymbol{\Omega}L_1T_L + \frac{1}{2}T_L^*L_1\boldsymbol{\Omega}T_L\\
&= \frac{1}{2}T_L^*\left(L_1\boldsymbol{\Omega} - \boldsymbol{\Omega}L_1\right)T_L\\
&= T_L^*\left(\vec{\Omega}\wedge\vec{n}_1 + \epsilon\left(\vec{\Omega}\wedge\vec{m}_1\right)\right)T_L
\end{aligned}
$$

where $L_1 = \mathbf{n}_1 + \epsilon\mathbf{m}_1 = (R^*T_LT^*)L_0(TT_L^*R)$.