Robotics and Computer–Integrated Manufacturing 52 (2018) 100–110

Contents lists available at ScienceDirect

Robotics and Computer–Integrated Manufacturing

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

Full length Article

Set-point control of robot end-effector pose using dual quaternion feedback[☆]

H.L. Pham[a], B.V. Adorno[b,*], V. Perdereau[a], P. Fraisse[c]

[a] UPMC Univ Paris 06, UMR 7222, ISIR, Paris, F-75005, France
[b] Universidade Federal de Minas Gerais, Department of Electrical Engineering, Belo Horizonte 31270-010 Minas Gerais, Brazil
[c] Université Montpellier 2, UMR 5506, LIRMM, Montpellier, 34095, France

ARTICLE INFO

Keywords:
Pose control
Dual quaternion
Robot manipulator

ABSTRACT

This paper proposes an easy-to-implement set-point control of robot end-effector pose using dual quaternion representation. A dual quaternion error invariant to the choice of the reference coordinate system is defined and the stability of two different kinds of controllers, one based on a constrained-Jacobian transpose and the other on a constrained-Jacobian pseudoinverse, both derived in dual quaternion space, is proved using Lyapunov theory. In addition, this paper describes a simple method to tune the proposed controllers from a practical and pragmatic point of view. Experiments with a 6-axis industrial robot are shown in order to highlight the efficiency of the method and the performance of the developed controllers.

1. Introduction

Whenever a robot arm is involved in an industrial process or any other professional, domestic or leisure activity, its dedicated role is always to move its end-effector so as to perform some manipulation tasks, as simple as pick-and-place operations or more complex ones, like using a tool or manipulating an object. The gripper, or possibly the hand attached to the end-effector, can be viewed as an actuator maneuvering the pose (i.e., the position and orientation) of the object/tool. In any case, the end-effector must be efficiently controlled and the task is typically described in terms of the desired pose of a frame attached to the end-effector.

In order to find the proper actuator commands that will make the robot arm succeed in the execution of the task, two main kinds of control schemes have been widely used depending whether the problem is solved in joint space [1] or in task-space [2]. Joint space control requires the solution of the inverse kinematics to compute the joint set-points corresponding to the desired end-effector pose, and the controller thus aims to ensure that joints attain the desired values. This solution is suitable for simple tasks in free space with predefined trajectories. Task-space control, in contrast, does not need the inverse kinematics to define the set-points but involve the kinematic transformations inside the control loop to generate the joint commands. This solution has been widely used in robotics, starting with Whitney's work [3], where a Cartesian velocity control was developed for a non-redundant manipulator. Liégeois [4] extended Whitney's idea in order to perform a secondary task in the null-space of the primary one. In Khatib's operational space [2], the Jacobian matrix played an important role for the dynamic control of the manipulator, and still today Jacobian-based control methods are object of intense research [5]. However, the controller design is more complex in task space than in joint space. The main reason for that is that the choice of a suitable set of variables for the task description is not always straightforward and necessarily involves kinematic transformations inside the control loop. These transformations make the controller gains dependent on the arm configuration so that a similar dynamic behavior is hardly ensured wherever the manipulator moves in the workspace.

In fact, while the position of the end-effector is widely and easily expressed in terms of three Cartesian coordinates, different representations are commonly adopted for the orientation. A minimal representation of the latter can be obtained with three parameters, for example the Euler angles. Despite its popularity, this representation suffers from inconsistency with the task geometry and representation singularities [6]. There are other parameterizations for the orientation, however, that does not suffer from singularities but use a non-minimal set of parameters (e.g., rotation matrices and unit quaternions) [7]. Moreover, position and orientation can be combined in a single representation—e.g., homogeneous transformation matrices (HTM) and dual quaternions—in order to represent rigid motions. For a more comprehensive list of different

☆ This paper was presented in part at the International Conference on Intelligent Robots and Systems, Taipei, Taiwan, October 2010.

* Corresponding author.
E-mail addresses: hoang-lan.pham@upmc.fr (H.L. Pham), adorno@ufmg.br (B.V. Adorno), veronique.perdereau@upmc.fr (V. Perdereau), fraisse@lirmm.fr (P. Fraisse).

https://doi.org/10.1016/j.rcim.2017.11.003
Received 25 May 2016; Received in revised form 30 August 2017; Accepted 4 November 2017
Available online 15 November 2017
0736-5845/

representations for rotations and translations, please refer to Shoham and Jen [8] and Chaturvedi et al. [9].

Regarding the orientation representation, there is an additional issue. Differently from the position error, which can be easily computed as a simple vector difference, the orientation error is computed in different ways depending on the parameterization used for the orientation. For that reason, some pose control solutions proposed in the literature are made of two separate control loops, one for the position and other for the orientation [10–12]. Some approaches, however, define different error equations for orientation and position, but group them together in a single control loop [13]. Recognizing that the aim of pose control is to move the robot from its initial pose to an arbitrary final one, other works use HTM to represent the pose and then define a suitable pose error directly on $SE(3)$, which gives a measure of the deviation of the actual pose from the desired one [14].

Instead of using HTM for representing the robot pose and rigid motions in general, other authors have chosen to use alternative representations, most notably unit dual quaternions. As HTM, which are elements of the group $SE(3)$, unit dual quaternions form an algebraic structure (more specifically, the group $\mathrm{Spin}(3)\ltimes\mathbb{R}^3$, which double covers the group $SE(3)$ [15]) and do not suffer from representation singularities, but possess a non-minimal number of parameters, as HTM. Some authors consider unit dual quaternions as the most efficient and compact tools to describe rigid transformations [16–19]; for instance, HTM has sixteen elements whereas dual quaternions has eight elements and dual quaternions multiplications are less expensive than HTM multiplications [20, p. 42]. In addition, dual quaternions have strong algebraic properties and can be used to represent rigid motions, twists, wrenches and several geometrical primitives—e.g., Plücker lines, planes, etc.—in a very straightforward way [21,22]. Moreover, it is easy to extract geometric parameters from a given unit dual quaternion (translation, axis of rotation, angle of rotation). Also, unit dual quaternions are easily mapped into a vector structure, which can be particularly convenient when controlling a robot, as there is no need to extract parameters from the dual quaternion to perform such task [23].

Several works have recently been focused on robot control using dual quaternions. For instance, Zhang et al. [24] used dual quaternions to develop a variable structure controller applied to omnidirectional robots, whereas Wang et al. [25] developed a distributed control law for coordinating several quad-rotors. Both Figueredo et al. [26] and Marinho et al. [27] proposed robust and optimal kinematic controllers, respectively, for robot manipulators based on dual quaternion representation. Wang et al. [28] proposed a generalized proportional control law based on unit dual quaternions applied to kinematic control of quad-rotors, and Han et al. [29] performed kinematic control applied to omnidirectional robots. Wang et al. [30] considered the relative coupled dynamics of two spacecrafts in order to tackle the rendezvous problem and Wang and Yu [31] developed a solution for the pose tracking problem of rigid bodies based on dual quaternions and the feedback linearization principle. The next subsection summarizes our paper contributions with respect to those aforementioned works.

1.1. Statement of contributions and organization of the paper

We propose an easy-to-implement set-point control of robot end-effector pose using dual quaternion representation. Differently from the vast majority of other approaches presented in the literature, which deal with the control of free-flying rigid bodies, our method directly relates the control inputs (i.e., joint velocities) to the unit dual quaternion representing the manipulator's end-effector pose in a very straightforward manner. This way, the task is defined directly in the task-space but the control inputs are given directly in the joint space, avoiding the use of an explicit inverse kinematics algorithm. In addition, we describe a simple method to tune the proposed controllers from a practical and pragmatic point of view, taking into account a second order dynamics for the robot joints, resulting in a method that is not only technical sound, but also useful for practical applications, mainly in industrial settings.

Furthermore, a dual quaternion error invariant to the choice of the reference coordinate system is defined and the stability of two different kinds of controllers, one based on a constrained-Jacobian transpose and the other on a constrained-Jacobian pseudoinverse, both derived in dual quaternion space, is proved using Lyapunov theory. Differently from previous approaches, we provide a thorough analysis of the analytical dual quaternion Jacobian matrix (i.e., the matrix that relates the joint velocities to the derivative of the unit dual quaternion representing the end-effector pose) in order to provide the conditions for the controller stability. This characterization is specially important because the dual quaternion Jacobian matrix is *always* singular by construction, even for completely actuated robots, as shown in Proposition 2. This fact has been constantly neglected in the literature, but our paper rigorously shows that the controller is stable even if the Jacobian matrix is always singular, as long as its rank remain constant.

The paper is organized as follows. Section 2 presents the mathematical background and the notation used in this paper, and develops the forward kinematics model (FKM) and the differential FKM in dual quaternion space. Section 3 first introduces a suitable error for the pose error and then describes the proposed control laws and presents their stability. Section 4 proposes a simple methodology for tuning the proposed controllers and Section 5 presents the results of experiments performed with a 6-axis Adept viper s850 robot. Last, Section 6 closes the paper with the conclusions and final remarks.

2. Mathematical background

Let $\hat{\imath},\hat{\jmath},\hat{k}$ be the three quaternionic units such that $\hat{\imath}^2=\hat{\jmath}^2=\hat{k}^2=\hat{\imath}\hat{\jmath}\hat{k}=-1$ [15]. The set of quaternions, which is an extension of the set of complex numbers, is given by

$$\mathbb{H}\triangleq\left\{a+b\hat{\imath}+c\hat{\jmath}+d\hat{k}\;:\;a,b,c,d\in\mathbb{R}\right\}.$$

Given $\boldsymbol{h}=a+b\hat{\imath}+c\hat{\jmath}+d\hat{k}$, its real and imaginary components are defined analogously to complex numbers as $\mathrm{Re}(\boldsymbol{h})\triangleq a$ and $\mathrm{Im}(\boldsymbol{h})\triangleq b\hat{\imath}+c\hat{\jmath}+d\hat{k}$, respectively.[1] The quaternion conjugate is defined as $\boldsymbol{h}^*\triangleq\mathrm{Re}(\boldsymbol{h})-\mathrm{Im}(\boldsymbol{h})$ and the quaternion norm is defined as $\|\boldsymbol{h}\|\triangleq\sqrt{\boldsymbol{h}^*\boldsymbol{h}}$. The set of pure quaternions is

$$\mathbb{H}_p\triangleq\{\boldsymbol{h}\in\mathbb{H}\;:\;\mathrm{Re}(\boldsymbol{h})=0\}.$$

Since $\mathbb{H}_p$ is isomorphic to $\mathbb{R}^3$ under the addition operation, pure quaternions behave as elements of $\mathbb{R}^3$. For instance, the translation given by the vector $\begin{bmatrix}x & y & z\end{bmatrix}^T$ is equivalent to the pure quaternion $x\hat{\imath}+y\hat{\jmath}+z\hat{k}$.

A unit quaternion (i.e., a quaternion with unit norm) is given by $\boldsymbol{r}=\cos(\phi/2)+\sin(\phi/2)\boldsymbol{n}$, where ϕ is a rotation angle around a rotation axis $\boldsymbol{n}=n_x\hat{\imath}+n_y\hat{\jmath}+n_z\hat{k}$ [32]. Unit quaternions under the multiplication operation belong to the group of rotations Spin(3), which double covers $SO(3)$ [15]. For unit quaternions, the inverse operation is given by the conjugate $\boldsymbol{r}^*=\cos(\phi/2)-\sin(\phi/2)\boldsymbol{n}$, such that $\boldsymbol{r}\boldsymbol{r}^*=\boldsymbol{r}^*\boldsymbol{r}=1$. Consequently, the group identity is the real number 1, since $1\boldsymbol{r}=\boldsymbol{r}1=\boldsymbol{r}$.

Dual quaternions extend the algebra of quaternions by the introduction of the dual unit ε (also known as Clifford unit), which has the following properties $\varepsilon\neq0$ but $\varepsilon^2=0$ [15]. The set of dual quaternions is thus defined as

$$\mathcal{H}\triangleq\{\boldsymbol{a}+\varepsilon\boldsymbol{b}\;:\;\boldsymbol{a},\boldsymbol{b}\in\mathbb{H}\}.$$

Let $\underline{\boldsymbol{h}}\in\mathcal{H}$ such that $\underline{\boldsymbol{h}}=\boldsymbol{a}+\varepsilon\boldsymbol{b}$. The quaternion not multiplied by ε is usually called the primary part and the one multiplied by ε is the dual part of a dual quaternion. Those terms are retrieved by using the operators [33]

$$\mathcal{P}\left(\underline{\boldsymbol{h}}\right)\triangleq\boldsymbol{a},\qquad\qquad\mathcal{D}\left(\underline{\boldsymbol{h}}\right)\triangleq\boldsymbol{b}.\tag{1}$$

[1] Some authors often refer to the quaternion components as scalar and vector parts since they use the scalar-plus-vector notation. Since we are using the hypercomplex notation, we extend the terminology of complex numbers to quaternions.