Gravity Compensation with Dual Quaternions

Francisco Jesús ARJONILLA GARCÍA[1] and Yuichi KOBAYASHI[1]

***Abstract*— Static balancing is essential for effective control of robots. The typical approach to obtain the gravity compensation terms is to construct an ad-hoc trigonometric model of the robot. This paper proposes a dual-quaternion representation of centroids that enables streamlined calculation of gravity compensation terms on rigid robots with arbitrary kinematic trees and arbitrary base orientations, valid for revolute and prismatic joints in kinematic chains with rigid links. The method was successfully tested on a six degrees-of-freedom manipulator model and evaluated against third-party software.**

I. Introduction

Gravity compensation is the neutralization of torques and forces induced by gravity in a robotic system. The solution to this well-known problem involves solving the equation

$$\frac{\partial \sum E_p}{\partial q_i} = 0 \tag{1}$$

for each $i = 1, \ldots, n$, where $\sum E_p$ is the summation of all potential energy, including gravitational, springs, magnetic, *etc.*, and q_i are the n generalized coordinates of the robot configuration [14]. When (1) follows, we say that the mechanism is statically balanced.

There are many advantages to neutralizing the effects of gravity. It has been shown that robotic manipulators are stable with PD controllers if the forces of gravity are neutralized beforehand [7][11]. In practice, PD controllers are complemented with an integrative term, *i.e.* a PID controller, to offset inaccuracies of mass value and location of the robot links and payload. Furthermore, in robot teaching by demonstration, gravity compensation facilitates external force following, *e.g.* the operator only needs to overcome the inertia of the links to change the pose of the robot. Cobots also benefit from gravity compensation by adding safety in case of collision. Other applications of gravity compensation in robotics include, but are not limited to, compliance robotics and torque-limited robots.

There are two broad categories of methods that implement gravity compensation [2][5]. Passive methods use ingenious mechanical arrangements involving springs, counterweights, magnets, etc. to sustain total potential energy in the system regardless of joint coordinates [4]. Conversely, active methods use auxiliary actuators, or even the main actuators, to produce forces and torques that counteract those exerted by gravity. Passive methods are more energy efficient, whereas active methods are more flexible [12].

There is an increasing number of low-budget serial manipulators that rely on active electric motors for gravity compensation. These robots require real-time calculation of gravity terms. More often than not, the solution involves solving trigonometric equations specific to each manipulator [15][20][21], in general under the assumption that the robot is installed in some specific orientation. However, trigonometry-based approaches are not well-suited to application-specific robots, such as snake-like robots with a high number of joints, or modular robots whose kinematic structure is not fully known during design.

In this paper we are concerned about the problem of calculating torques and forces to be applied by the gravity-compensating device. Inertial forces and other dynamic effects are not considered because they do not affect static balancing. The central problem is finding the right coordinate transformation function to express relevant centroids and gravity vector in the same coordinate system, then operate to obtain the gravity terms.

The solution to this problem is well known, but here we propose a streamlined method to solve the gravity compensation problem based on dual-quaternion algebra [6][8]. We show that an extension to the representation of points in dual-quaternion form with added support for mass values is closed under dual-quaternion algebra. This extension is then developed into an efficient and general method to calculate the gravity-balancing forces and torques of rigid robots. Quaternions and dual quaternions offer several advantages over homogeneous matrices, namely numerical stability, compactness and computational efficiency [23].

II. Algebra review

In this section we briefly review the quaternion and dual-quaternion algebra used in the rest of the paper.

A. Quaternions

Quaternions are a non-commutative extension to complex numbers with three distinct complex bases i, j and k that follow the axioms

$$\vec{\mathrm{i}}^{\,2} = \vec{\mathrm{j}}^{\,2} = \vec{\mathrm{k}}^{\,2} = \vec{\mathrm{i}}\,\vec{\mathrm{j}}\,\vec{\mathrm{k}} = -1, \tag{2}$$

Given a *rotation* defined by an angle θ around the unitary vector $\vec{r}$, it is representable as a unit quaternion $\boldsymbol{r}$ by

$$\boldsymbol{r} = s_r + \vec{v}_r = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\,\vec{r}. \tag{3}$$

$\boldsymbol{r}$ and $-\boldsymbol{r}$ represent the same rotation.

Rotations in quaternionic form rotate points $\boldsymbol{p} = \vec{p}$ by the operation

$$\boldsymbol{p}\,' = \boldsymbol{r}\,\boldsymbol{p}\,\boldsymbol{r}^{\,*}, \tag{4}$$

[1]Department of Mechanical Engineering, Shizuoka University, 3-5-1 Johoku, Chuo-ku, Hamamatsu, 432-8561 Shizuoka, Japan arjonilla.francisco@shizuoka.ac.jp

TABLE I

MATHEMATICAL NOTATION FOLLOWED THROUGHOUT THE PAPER.

Name	Notation	Element of	Construction
Scalar	s	$\mathbb{R}$	
Vector	$\vec{v}$	$\mathbb{R}^3$	$\vec{v} = v_x\,\vec{\mathrm{i}} + v_y\,\vec{\mathrm{j}} + v_z\,\vec{\mathrm{k}}$
Matrix	M	$\mathbb{R}^{m\times n}$	
Coordinate system	O_i		
Quaternion	$\boldsymbol{q}$	$\mathbb{H} \cong \mathbb{R}^4$	$\boldsymbol{q} = s + \vec{v}$
Dual quaternion	$\boldsymbol{Q}$	$\mathbb{H}^2$	$\boldsymbol{Q} = \boldsymbol{r} + \varepsilon\boldsymbol{p}$

which reduces to Rodrigues' rotation formula [19]. $\boldsymbol{r}^* = s_r - \vec{v}_r$ is the conjugate of $\boldsymbol{r}$ and also its inverse because it is unitary: $\boldsymbol{r}^*\boldsymbol{r} = \boldsymbol{r}\boldsymbol{r}^* = 1$.

B. Dual quaternions

Dual quaternions add the dual base ε to quaternions, where $\varepsilon^2 = 0$:

$$\boldsymbol{Q} = \boldsymbol{r} + \varepsilon\,\boldsymbol{p}, \tag{5}$$

where $\text{direct}\,(\boldsymbol{Q}) = \boldsymbol{r}$ is the *direct part* and $\text{dual}\,(\boldsymbol{Q}) = \boldsymbol{p}$ is the *dual part*. $\boldsymbol{r}$ and $\boldsymbol{p}$ are quaternions.

Similarly to quaternions, dual quaternions are commutative and associative under addition, but non-commutative under multiplication:

$$\boldsymbol{Q}_1\,\boldsymbol{Q}_2 = \boldsymbol{r}_1\boldsymbol{r}_2 + \varepsilon\,(\boldsymbol{r}_1\boldsymbol{p}_2 + \boldsymbol{p}_1\boldsymbol{r}_2) \tag{6}$$

with $\varepsilon^2\,(\boldsymbol{p}_1\boldsymbol{p}_2) = 0$. The conjugate of a dual quaternion $\boldsymbol{Q}$ is defined as the conjugate of the direct and dual parts

$$\boldsymbol{Q}^* = \boldsymbol{r}^* + \varepsilon\,\boldsymbol{p}^*. \tag{7}$$

C. Points and transformations

A *point* $\vec{p}$ is represented in dual-quaternion form by

$$\boldsymbol{P} = 1 + \varepsilon\,\boldsymbol{p} = 1 + \varepsilon\,\vec{p}. \tag{8}$$

However, a *translation* defined by vector $\vec{t}$ takes the following form:

$$\boldsymbol{T} = 1 + \varepsilon\tfrac{1}{2}\vec{t}. \tag{9}$$

Pure rotations $\boldsymbol{R} = \boldsymbol{r}$ have no dual part.

Rigid-body transformations are obtained by composing a translation with a rotation, so

$$\boldsymbol{B} = \boldsymbol{T}\boldsymbol{R} = \left(1 + \varepsilon\tfrac{1}{2}\vec{t}\right)\boldsymbol{r} = \boldsymbol{r} + \varepsilon\tfrac{1}{2}\vec{t}\,\boldsymbol{r}. \tag{10}$$

The rotation component of $\boldsymbol{B}$ is obtained by $\boldsymbol{r} = \text{direct}\,(\boldsymbol{B})$ and the translation component by $\vec{t} = 2\,\text{dual}\,(\boldsymbol{B})\,\boldsymbol{r}^*$.

The *inverse* of a rigid body transformation $\boldsymbol{B}$ is:

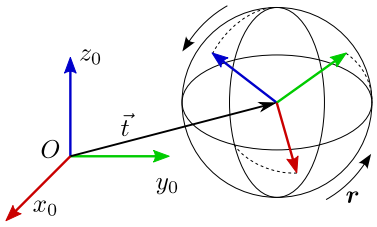
$$\boldsymbol{B}^{-1} = \boldsymbol{B}^*. \tag{11}$$

Composition of rigid body transformations is

$$\begin{aligned}\boldsymbol{B}_{21} &= \boldsymbol{B}_2\boldsymbol{B}_1\\ &= \boldsymbol{r}_2\boldsymbol{r}_1 + \varepsilon\tfrac{1}{2}\left(\boldsymbol{r}_2\vec{t}_1\,\boldsymbol{r}_1 + \vec{t}_2\,\boldsymbol{r}_2\boldsymbol{r}_1\right)\\ &= \boldsymbol{r}_{21} + \varepsilon\tfrac{1}{2}\left(\vec{t}_2 + \boldsymbol{r}_2\vec{t}_1\,\boldsymbol{r}_2^*\right)\boldsymbol{r}_{21},\end{aligned} \tag{12}$$

where transformation $\boldsymbol{B}_2$ follows $\boldsymbol{B}_1$ and $\boldsymbol{r}_{21} = \boldsymbol{r}_2\boldsymbol{r}_1$.

Fig. 1. Geometric interpretation of a rigid body transformation $\boldsymbol{Q} = \boldsymbol{r} + \varepsilon\tfrac{1}{2}\vec{t}\,\boldsymbol{r}$ in dual-quaternion form.



A rigid-body transformation of point $\boldsymbol{P}$ is reduced to

$$\begin{aligned}\boldsymbol{P}' &= \boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^{-*},\\ &= \boldsymbol{r}\boldsymbol{r}^* + \varepsilon\left(\tfrac{1}{2}\vec{t}\,\boldsymbol{r}\boldsymbol{r}^* + \boldsymbol{r}\vec{p}\,\boldsymbol{r}^* - \tfrac{1}{2}\boldsymbol{r}\left(\vec{t}\,\boldsymbol{r}\right)^*\right)\\ &= 1 + \varepsilon\left(\vec{t} + \boldsymbol{r}\vec{p}\,\boldsymbol{r}^*\right),\end{aligned} \tag{13}$$

where the conjugate of a vector is its negation $\vec{t}^{\,*} = -\vec{t}$ and $\boldsymbol{B}^{-*}$ is the conjugate of $\boldsymbol{B} = \boldsymbol{r} + \varepsilon\,\boldsymbol{p}$ with a negated dual part: $\boldsymbol{B}^{-*} = \boldsymbol{r}^* - \varepsilon\,\boldsymbol{p}^*$. $\boldsymbol{P}'$ is indeed a point in dual-quaternion form with $\vec{p}\,' = \vec{t} + \boldsymbol{r}\vec{p}\,\boldsymbol{r}^*$.

Refactoring (13) results in the *inverse transformation* of $\boldsymbol{P}'$:

$$\boldsymbol{P} = (\boldsymbol{B})^{-1}\,\boldsymbol{P}'\,\left(\boldsymbol{B}^{-*}\right)^{-1} = \boldsymbol{B}^*\boldsymbol{P}'\,\boldsymbol{B}^- \tag{14}$$

where $\boldsymbol{B}^- = \boldsymbol{r} - \varepsilon\tfrac{1}{2}\vec{t}\,\boldsymbol{r}$.

III. GRAVITY COMPENSATION

In this section we extend the dual-quaternion representation of rigid-body transformations and points to incorporate representations of centroids.

We consider a robotic manipulator consisting of a sequence of rigid bodies, or links, where each link can rotate or slide along an axis fixed with respect to the preceding link. The force of gravity exerts a constant force to each link. The mass of each link can be reduced to a single point located at the center of mass, or centroid. There is no need to consider moments of inertia of the links or other dynamic effects because gravity is a static force.

A. Centroids

The following proposition is the core contribution of this paper.

Proposition 1: The representation of a centroid C of mass m located at $\boldsymbol{P}_C = 1 + \varepsilon\,\vec{c}$ in dual-quaternion form is

$$\boldsymbol{C} = m\boldsymbol{P}_C = m + \varepsilon\,m\vec{c}. \tag{15}$$

Proof: The following properties show that $\boldsymbol{C}$ is closed under addition and under rigid-body transformation. The mass of a centroid is the direct part of the centroid:

$$m = \text{direct}\,(\boldsymbol{C}) \tag{16}$$

and the location of the centroid is

$$\vec{c} = \frac{\text{dual}\,(\boldsymbol{C})}{\text{direct}\,(\boldsymbol{C})}. \tag{17}$$

The norm of $\boldsymbol{C}$ is also the mass of the centroid, thus a centroid is not unitary:

$$\|\boldsymbol{C}\| = \|m\boldsymbol{P}\| = m\,\|\boldsymbol{P}\| = m. \tag{18}$$

Centroids can be added to each other when they are expressed in the same coordinate system.

$$\boldsymbol{C}_{12} = \boldsymbol{C}_1 + \boldsymbol{C}_2 = (m_1 + m_2) + \varepsilon\,(m_1\vec{c}_1 + m_2\vec{c}_2) \tag{19}$$

The resulting centroid $\boldsymbol{C}_{12}$ merges centroids $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. Indeed, the mass of $\boldsymbol{C}_{12}$ is $m_{12} = \text{direct}\,(\boldsymbol{C}) = m_1 + m_2$ and its location is the weighted average of the added centroids

$$\vec{c}_{12} = \frac{\text{dual}\,(\boldsymbol{C}_{12})}{\text{direct}\,(\boldsymbol{C}_{12})} = \frac{m_1\vec{c}_1 + m_2\vec{c}_2}{m_1 + m_2}. \tag{20}$$

Addition of centroids is associative and commutative because addition of dual quaternions is so, hence the centroid of a set of rigid bodies $1, 2, \ldots, n$ with individual centroids $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_n$ is simply

$$\boldsymbol{C} = \sum_{i=1}^{n} \boldsymbol{C}_i \tag{21}$$

where $\boldsymbol{C}_i = m_i + \varepsilon\, m_i\vec{c}_i$. Translation of centroids is similar to translations of points:

$$\begin{aligned}\boldsymbol{C}_T &= \boldsymbol{T}\boldsymbol{C}\boldsymbol{T}^{-*} = \boldsymbol{T}m\boldsymbol{P}\boldsymbol{T}^{-*} = m\left(\boldsymbol{T}\boldsymbol{P}\boldsymbol{T}^{-*}\right)\\ &= m + \varepsilon\, m\left(\vec{t} + \vec{c}\right) = m + \varepsilon\left(m\vec{t} + \text{dual}\,(\boldsymbol{C})\right)\\ &= \boldsymbol{C} + \varepsilon\, m\vec{t},\end{aligned} \tag{22}$$

with $\boldsymbol{T} = 1 + \varepsilon\frac{1}{2}\vec{t}$. Similarly, rotation of centroids operate like rotation of points:

$$\begin{aligned}\boldsymbol{C}_R &= \boldsymbol{R}\boldsymbol{C}\boldsymbol{R}^{-*} = m\left(\boldsymbol{R}\boldsymbol{P}\boldsymbol{R}^{-*}\right)\\ &= m + \varepsilon\, m\left(\boldsymbol{r}\vec{c}\boldsymbol{r}^*\right) = m + \varepsilon\,\boldsymbol{r}\,\text{dual}\,(\boldsymbol{C})\,\boldsymbol{r}^*.\end{aligned} \tag{23}$$

Thus, a rigid-body transformation $\boldsymbol{B} = \boldsymbol{T}\boldsymbol{R} = \boldsymbol{r} + \varepsilon\frac{1}{2}\vec{t}\boldsymbol{r}$ applied to the centroid $\boldsymbol{C}$ is:

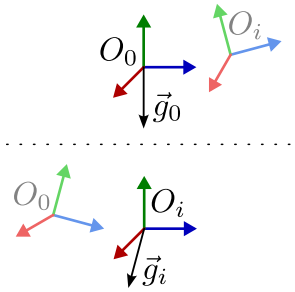
$$\begin{aligned}\boldsymbol{C}_B &= \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^{-*} = m\left(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^{-*}\right) = m + \varepsilon\, m\left(\vec{t} + \boldsymbol{r}\vec{c}\boldsymbol{r}^*\right)\\ &= m + \varepsilon\left(m\vec{t} + \boldsymbol{r}\,\text{dual}\,(\boldsymbol{C})\,\boldsymbol{r}^*\right).\end{aligned} \tag{24}$$

As expected, only the location is transformed, not the mass. Therefore, centroids as dual quaternions are closed under addition and rigid body transformation. ■

The physical units of a centroid are, for the direct part, $[\text{direct}\,(\boldsymbol{C})] = \text{M}$, and for the dual part, $[\text{dual}\,(\boldsymbol{C})] = \text{ML}$. That is, $[\boldsymbol{C}] = \text{M} + \varepsilon\,\text{ML}$. The units hold under rigid body transformations:

$$\begin{aligned}[\boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^{-1}] &= (1 + \varepsilon\,\text{L})(\text{M} + \varepsilon\,\text{ML})(1 + \varepsilon\,\text{L})\\ &= \text{M} + \varepsilon\,(\text{LM} + \text{ML} + \text{ML})\\ &= \text{M} + \varepsilon\,\text{ML}.\end{aligned} \tag{25}$$

Fig. 2. Transformation of the gravity vector $\vec{g}_0$ (top) adjusts the local direction of the vector relative to the coordinate system O_i (bottom).



B. Torques and forces

Heretofore, we have presented an algebra to deal with centroid conversion between coordinate systems by rigid body transformations. In this section we deduce the equations to obtain the torques and forces necessary for static balancing in revolute and prismatic joints.

Let a coordinate system be fixed to a rigid body $i-1$ such that the z-axis is aligned with the force of gravity $\vec{g}_{i-1} = g\,\vec{\text{k}}$, where $g \simeq -9.81\,\text{m/s}^2$. Consider another coordinate system for the articulated link i whose z-axis is aligned to the axis of rotation (revolute joints) or sliding direction (prismatic joints) of the joint. Let $\boldsymbol{B}_i = \boldsymbol{r}_i + \varepsilon\,\boldsymbol{p}_i$ be the transformation in dual-quaternion form between these two coordinate systems, such that any point $\vec{p}_i$ expressed in the link coordinate system is expressed in the coordinate system by

$$\vec{p}_{i-1} = \boldsymbol{B}_i\,\vec{p}_i\,\boldsymbol{B}_i^{-*}. \tag{26}$$

Now, let $\boldsymbol{C}_i = m_i + \varepsilon\, m_i\vec{c}_i$ be the centroid of the articulated link in dual-quaternion form in the coordinate system of the articulated link. Then, we can transform the gravity vector $\vec{g}_{i-1}$ to this coordinate system (Figure 2) by

$$\vec{g}_i = \boldsymbol{r}_i^*\,\vec{g}_{i-1}\,\boldsymbol{r}_i^* \tag{27}$$

in order to obtain the torque τ_i (for revolute joints) or force f_i (for prismatic joints) required to counteract gravity, which are equal in magnitude and opposite to the force produced by gravity:

$$\tau_i = -m_i\,(\vec{c}_i \times \vec{g}_i)\cdot\vec{\text{k}} = -\,(\text{dual}\,(\boldsymbol{C}_i) \times \vec{g}_i)\cdot\vec{\text{k}} \tag{28}$$

$$f_i = -m_i\vec{g}_i\cdot\vec{\text{k}} = -\text{direct}\,(\boldsymbol{C}_i)\,\vec{g}_i\cdot\vec{\text{k}} \tag{29}$$

Note that $\text{dual}\,(\boldsymbol{C}_i)$ accounts for both the position vector and the mass of the centroid in equation (28). The inner product with $\vec{\text{k}}$ realizes projection of force or torque into the axis of the joint, which we assumed to be coincident with the z-axis.

IV. Serial Robots

In this section we streamline centroids in dual-quaternion form to calculate static-balancing forces and torques on serial robots specified by coordinate transformation functions between link coordinate systems.

A. Streamlining operations

Consider a kinematic chain with n joints (revolute or prismatic) and n links. The problem we face is to streamline the calculation of torques and forces (28, 29) that counteract the gravity forces of each joint $i \in \{1, \ldots, n\}$ in the robot.

Let us assign a fixed coordinate system O_i to each link i, with O_0 corresponding to the global coordinate system. The coordinate transformation function from link n to link $n-1$ is $\boldsymbol{B}_n = \boldsymbol{r}_n + \varepsilon \frac{1}{2} \vec{t}_n \boldsymbol{r}_i$. and the centroid of link n in O_n by $\boldsymbol{C}_n$. Then, we know that $\boldsymbol{C}_n$ in coordinate system O_{n-1} is

$$\boldsymbol{B}_n \boldsymbol{C}_n \boldsymbol{B}_n^{-*}. \tag{30}$$

Now we add the centroid of link $n-1$ to obtain the accumulated centroid $\boldsymbol{A}_{n-1}$ of links $n-1$ and n in O_{n-1}:

$$\boldsymbol{A}_{n-1} = \boldsymbol{C}_{n-1} + \boldsymbol{B}_n \boldsymbol{C}_n \boldsymbol{B}_n^{-*}. \tag{31}$$

Using this reasoning, we iteratively define the accumulated centroid of all links from link $i < n$ to link n by

$$\boldsymbol{A}_i = \boldsymbol{C}_i + \boldsymbol{B}_{i+1} \boldsymbol{A}_{i+1} \boldsymbol{B}_{i+1}^{-*} \tag{32}$$

and $\boldsymbol{A}_n = \boldsymbol{C}_n + \boldsymbol{C}_p$, where $\boldsymbol{C}_p$ is the centroid of the payload in $\boldsymbol{O}_n$. When a kinematic tree branches at link i, the centroid of each branch is added to $\boldsymbol{A}_i$:

$$\boldsymbol{A}_i = \boldsymbol{C}_i + \boldsymbol{B}_{i+1} \boldsymbol{A}_{i+1} \boldsymbol{B}_{i+1}^{-*} + \boldsymbol{B}_{i+1}^{'} \boldsymbol{A}_{i+1}^{'} \boldsymbol{B}_{i+1}^{'-*} + \cdots . \tag{33}$$

(28) and (29) indicate that we need $\boldsymbol{A}_i$ and $\vec{g}_i$ to calculate gravity forces and torques, so the next step is to either transform the accumulated centroid of each link i to the global coordinate system, or transform the gravity vector to each coordinate system O_i. The latter is more efficient than transforming $\boldsymbol{A}_i$ to O_0 because transformation of vectors involve only rotations, so operating with the direct part $\boldsymbol{r}_i$ of $\boldsymbol{B}_i$ suffices to transform the gravity vector to O_i. Thus, on this occasion the iterative transformation goes from the base of the robot to link i. Assuming that the robot is installed in the predefined vertical orientation, then $\vec{g}_0 = \vec{g}$ and

$$\vec{g}_i = \boldsymbol{r}_i^* \vec{g}_{i-1} \boldsymbol{r}_i. \tag{34}$$

For convenience, let us assume that axis $\vec{\mathrm{k}}$ of O_i is located along of the axis of rotation or translation of joint i. Then, we can replace $\boldsymbol{A}_i$ and $\vec{g}_i$ straightforwardly in (28) and (29) to obtain the torque τ_i and force f_i required for static balance.

For reasons of computational efficiency, it is preferable to separate the transformation function defined by $\boldsymbol{B}_i$ in two components such that

$$\boldsymbol{B}_i = \boldsymbol{N}_i \boldsymbol{J}_i. \tag{35}$$

The first component $\boldsymbol{N}_i$ corresponds to the transformation function from link i to link $i-1$ when the robot is in the neutral pose. The second component $\boldsymbol{J}_i$ corresponds to the state of the joint. $\boldsymbol{N}_i$ is fixed by the kinematic design of the robot, so it only needs to be calculated once. $\boldsymbol{J}_i$ is a rotation only when the joint is revolute and has the equation

$$\boldsymbol{J}_i = \boldsymbol{R}_{\vec{\mathrm{k}}}(q_i) = \cos \frac{q_i}{2} + \sin \frac{q_i}{2} \vec{\mathrm{k}}. \tag{36}$$

Fig. 3. Matlab rendering of robot UR5 configured as specified on table II.

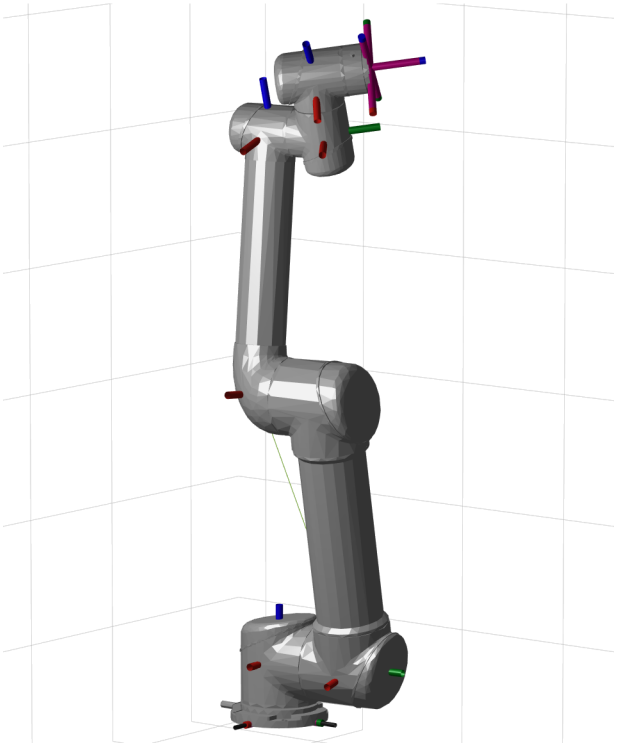


TABLE II

DENAVIT-HARTENBERG PARAMETERS OF ROBOT UR5 UNDER THE PROPOSED INDEX NOTATION. THESE PARAMETERS INDICATE THE STRUCTURE AND THE NEUTRAL POSE OF THE ROBOT, WHILST THE LAST COLUMN CONTAINS THE VALUES OF THE JOINT ROTATIONS USED IN THE EXAMPLE (SEE TEXT).

i	d_i [m]	θ_i [rad]	a_i [m]	α_i [rad]	q_i [rad]
1	–	–	–	–	0.1
2	l_1	π	0	$\frac{\pi}{2}$	0.2
3	0	$\frac{\pi}{2}$	l_2	π	0.3
4	0	0	l_3	π	0.4
5	l_4	$\frac{\pi}{2}$	0	$\frac{\pi}{2}$	0.5
6	l_5	π	0	$\frac{\pi}{2}$	0.6
e	l_6	0	0	0	–

We have now the accumulated centroid $\boldsymbol{A}_i$ and the gravity vector $\vec{g}_i$ expressed in the same coordinate system O_i; we can now calculate gravity compensating torques and forces using equations (28) and (29), respectively.

V. EVALUATION

In this section we apply centroids in dual-quaternion form to a six degrees-of-freedom serial manipulator to demonstrate the practical utility of the proposed framework and to validate the results.

A. Problem setting

The UR5 robot from manufacturer Universal Robots [22] is a six-degrees-of-freedom (6 DoF) robot with non-spherical

Fig. 4. Kinematics of robot UR5. Coordinate systems are placed according to the proposed index notation for D-H parameters.

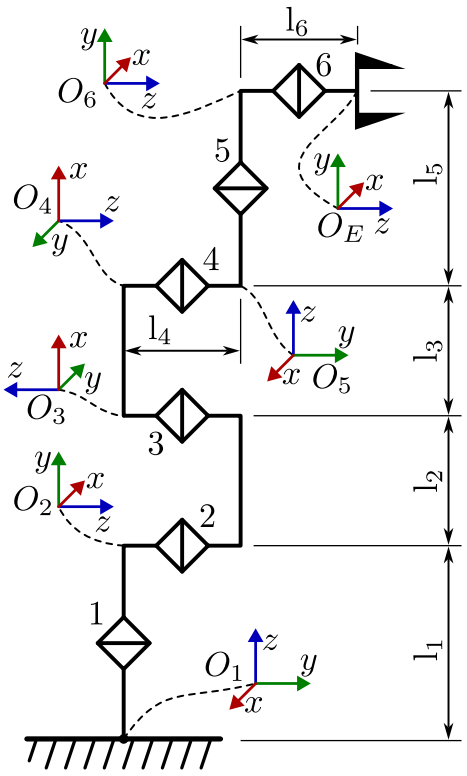


TABLE III

DIMENSIONAL AND CENTROID DATA OF ROBOT UR5 RELATIVE TO THE PROPOSED INDEX NOTATION.

i	l_i [m]	m_i [kg]	$\vec{c}_i$ [m]
1	0.089159	3.7	$(0, 0.00193, 0.063549)$
2	0.425	8.393	$(0, 0.2125, 0.11336)$
3	0.39225	2.33	$(0.24225, 0, -0.0265)$
4	0.10915	1.219	$(0.01634, 0, 0.10735)$
5	0.09465	1.219	$(0, 0.01634, 0.09285)$
6	0.0823	0.1879	$(0, 0, 0.081141)$

wrist (Figure 4). Table II shows the D-H parameters of the robot, which is in a non-neutral pose with joint values $q_i = 0.1i$ as depicted in figure 3. The length, mass and location of the center of mass of each link are those specified in table III. The problem is to calculate the torque required on the joints of this robot to neutralize the effect of gravity when it is installed horizontally on the floor with no payload.

B. Algorithm

Assuming that $O_0 \equiv O_1$, then $\boldsymbol{N}_1 = 1$ and the rest of the rigid-body transformations in neutral pose for each quadruplet of D-H parameters is

$$\begin{aligned}
\boldsymbol{N}_1 &= 1 \\
\boldsymbol{N}_2 &= \frac{\sqrt{2}}{2}\left(\vec{\mathrm{j}} + \vec{\mathrm{k}}\right) + \varepsilon\frac{\sqrt{2}}{4}\mathrm{l}_1\left(-1 - \vec{\mathrm{i}}\right) \\
\boldsymbol{N}_3 &= \frac{\sqrt{2}}{2}\left(\vec{\mathrm{i}} + \vec{\mathrm{j}}\right) + \varepsilon\frac{\sqrt{2}}{4}\mathrm{l}_2\left(-1 - \vec{\mathrm{k}}\right) \\
\boldsymbol{N}_4 &= \vec{\mathrm{i}} - \varepsilon\frac{1}{2}\mathrm{l}_3 \\
\boldsymbol{N}_5 &= \frac{1}{2}\left(1 + \vec{\mathrm{i}} + \vec{\mathrm{j}} + \vec{\mathrm{k}}\right) + \varepsilon\frac{1}{4}\mathrm{l}_4\left(-1 - \vec{\mathrm{i}} + \vec{\mathrm{j}} + \vec{\mathrm{k}}\right) \\
\boldsymbol{N}_6 &= \frac{\sqrt{2}}{2}\left(\vec{\mathrm{j}} + \vec{\mathrm{k}}\right) + \varepsilon\frac{\sqrt{2}}{4}\mathrm{l}_5\left(-1 - \vec{\mathrm{i}}\right). \qquad (37)
\end{aligned}$$

The transformation corresponding to the state of each joint is (36)

$$\boldsymbol{J}_i = \cos\frac{q_i}{2} + \sin\frac{q_i}{2}\,\vec{\mathrm{k}} = \cos 0.05i + \sin 0.05i\,\vec{\mathrm{k}}. \qquad (38)$$

Multiplying both transformations, we get the transformation functions between links $\boldsymbol{B}_i = \boldsymbol{N}_i\boldsymbol{J}_i$ (35). The centroids of each link in dual-quaternion form are $\boldsymbol{C}_i = m_i + \varepsilon\, m_i\vec{c}_i$ where m_i and $\vec{c}_i$ are specified in table III relative to the local coordinates O_i of each link. The accumulated centroids $\boldsymbol{A}_i$ are obtained from $\boldsymbol{B}_i$ and $\boldsymbol{C}_i$ by iterating (32) starting from $\boldsymbol{A}_6 = \boldsymbol{C}_6$. The gravity vector g_i at each coordinate system O_i is also obtained iteratively as shown in (34), starting from $\vec{g}_0 = (0, 0, -9.81)$. Substituting $\boldsymbol{A}_i$ and $\vec{g}_i$ in (28) yields the desired gravity-compensating torque values.

C. Experimental setup

The algorithm described in the previous section was implemented and executed in MATLAB. The link to the source code of this implementation is in the appendix.

The D-H index notation between the manufacturer's data [22] Matlab's model and our approach were different, albeit equally valid. Thus, the kinematic chains of the three conventions were incompatible, so centroid data had to be converted between the conventions. We used the POS library [17] to transform the centroid location from the manufacturer's data to Matlab's convention and to our convention.

The evaluation of centroids consisted in comparing the torque values obtained by our method to those given by the function `gravityTorque()` in Matlab's Robotics System Toolbox [13]. To that end, we used Matlab's model of the UR5 robot with masses and centers of mass modified to accurately reflect the manufacturer's data.

Our calculation results matched those of Matlab, showing the validity of our approach. Additionally, we made forward kinematic calculation tests in various joint states to assess the coordinate transformation representation with matching results in both the manufacturer's tool and Matlab's Robotic Systems Toolbox. The numeric results of the accumulated centroids, the gravity vectors and the torques are recorded in table IV.

VI. CONCLUSION

This paper has proposed the use of dual quaternions to calculate gravity-compensation forces and torques for robotic

TABLE IV

NUMERICAL RESULTS OF GRAVITY COMPENSATION EXAMPLE OF UR5 ROBOT.

i	$\boldsymbol{A}_i$ [kg+kgm]	$\vec{g}_i$ [ms^{-2}]	$\vec{\tau}_i$ [Nm]
1	$17.05 + \varepsilon\,(0.642, 1.336, 6.973)$	$(0, 0, -9.81)$	0
2	$13.35 + \varepsilon\,(0.473, 5..565, 1.328)$	$(-1.95, -9.61, 0)$	-6.29883
3	$4.96 + \varepsilon\,(1.740, -0.043, -0.377)$	$(-9.76, 0.98, 0)$	-1.28212
4	$2.626 + \varepsilon\,(0.151, -0.017, 0.315)$	$(-9.37, 2.90, 0)$	-0.27943
5	$1.407 + \varepsilon\,(0, 0.035, 0.131)$	$(2.54, -1.39, -9.37)$	0.089465
6	$0.188 + \varepsilon(0, 0, 0.015)$	$(-7.39, -6.30, -1.39)$	0

applications by extending dual-quaternion representation of centroid points to incorporate the centroid mass. The advantages of using dual quaternions over ad-hoc trigonometry and matrices is numerical stability, smaller memory footprint and a simplified algebra [23]. Our approach leverages these advantages in the calculation of gravity-compensating torques and forces of rigid robots. These results can be easily generalized to kinematic trees with cylindrical and screw joints, and can can be applied to offline calculation of passive devices such as balancing springs, and online calculation of active devices such as electric actuators.

The limitations of the proposed method are that centroid location and mass of each link and load must be known beforehand. We leave for future works the analysis of the inverse operation, that is, calculation of centroid from joint forces and torques, which is important in robots with unspecified loads that need to adapt automatically to payload changes in the end effector [9].

This research has explored new avenues in mathematical frameworks for mainstream robotics by introducing elements of Clifford algebras into common robotic calculations. We hope that the robotics community will benefit of leveraging the potential of Clifford algebras in the near future. Promising efforts include extending dual quaternions to calculation of robot dynamics.

APPENDIX

The source code is available at:
`http://sensor.eng.shizuoka.ac.jp/~parjonilla/downloads/IROS2025.zip`

REFERENCES

[1] B. V. Adorno, Robot kinematic modeling and control based on dual quaternion algebra — Part I: Fundamentals, Hal-01478225, 2017.

[2] V. Arakelian (ed.) Gravity compensation in robotics, Rennes, France, 2022.

[3] E. Bayro-Corrochano, Geometric algebra applications Vol. II: Robot modelling and control, Springer Charm, Guadalajara, Mexico, 2020.

[4] J. Boisclair, P.L. Richard, T. Laliberté and C. Gosselin, Gravity compensation of robotic manipulators using cylindrical Halbach arrays, IEEE/ASME Transactions on Mechatronics, 22(1): 457–464, 2017.

[5] Y.R. Chheta, T.M. Joshi, K.K. Gotewal and S.M. Manoah, A review on passive gravity compensation, in International Conference on Electronics, Communication and Aerospace Technology, Coimbatore, India, pp. 184–189, 2017.

[6] W. K. Clifford, Preliminary sketch of biquaternions, Proceedings of the London Mathematical Society, 1871, s1-4(I), pp. 381–395, 1978.

[7] J. Cvejn, M. Zapletal, Feedback control of robot manipulators by using gravity and inertial effects compensation, in International Carpathian Control Conference, Krakow-Wieliczka, Poland, 2019.

[8] W. R. Hamilton, On quaternions: on a new system of imaginaries in Algebra, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 25(169), pp. 489–495, 1844.

[9] A. A. Hayat, A. D. Udai and S. K. Saha, Kinematic chain of an industrial robot and its identification for gravity compensation, in International Conference on Machines and Mechanisms, IIT Roorkee, India, 2013.

[10] K. Kanatani, Understanding geometric algebra: Hamilton, Grassman, and Clifford for computer vision and graphics, CRC Press, Okayama, Japan, 2015.

[11] R. Kelly, V. Santibáñez Davila, A. Loría, Control of robot manipulators in joint space, Springer-Verlag, London, 2005.

[12] Klimchik A and Pashkevich A, Stiffness modelling for gravity compensators, in Gravity Compensation in Robotics, Rennes, France: Springer Cham, pp. 27–71, 2022.

[13] Mathworks, Matlab Robotics System Toolbox Documentation, `https://uk.matworks.com/help/robotics`, accessed July, 2024.

[14] G. Mottola, M. Cocconcelli, R. Rubini and M. Carricato, Gravity balancing of parallel robots by constant-force generators. In Gravity Compensation in Robotics, Springer Cham, pp. 229–273, Rennes, France, 2022.

[15] S. Moubarak, M. T. Pham, R. Moreau and T. Redarce, Gravity compensation of an upper extremity exoskeleton. in Annual International Conference of the IEEE Engineering in medicine and Biology Society, Buenos Aures, pp. 4489–4493, 2010.

[16] E. Pennestri and P. P. Valentini, Dual quaternions as a tool for rigid body motion analysis: a tutorial with an application to biomechanics, The Archive of Mechanical Engineering, 57(2), 2010.

[17] The POS library, URL `https://gitlab.com/ninbot/pos`, accessed July, 2024.

[18] L. A. Radavelli, A comparative study of the kinematics of robot manipulators by Denavit-Hartenberg and dual quaternion, Mecánica Computacional, 31, pp. 2833–2848, 2012.

[19] O. Rodrigues, Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire, Journal de Mathématiques Pures et Appliquées, 5, pp. 380–440, 1840.

[20] S. Shirata, A. Konno and M. Uchiyama, Design and evaluation of a gravity compensation mechanism for a humanoid robot, in IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, U.S.A., pp. 3635–3640, 2007.

[21] M. Uemura, Y. Mitabe and S. Kawamura, Simultaneous gravity and gripping force compensation mechanism for lightweight hand-arm robot with low-reduction reducer, Robotica, 37(6), pp. 1090–1103, 2019.

[22] Universal Robots – DH parameters for calculations of kinematics and dynamics. URL `https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics`, accessed June, 2024.

[23] X. Wang and H. Zhu, On the comparisons of unit dual quaternion and homogeneous transformation matrix, Advances in Applied Clifford Algebras, 24(1), pp. 213–229, 2014.