# Quaternions and Dual Quaternions: Singularity-Free Multirobot Formation Control

**Ignacio Mas** (ID) · **Christopher Kitts**

**Abstract** Cluster space control is a method of multirobot formation keeping that considers a group of robots to be a single entity, defining state variables to represent characteristics of the group, such as position, orientation, and shape. This technique, however, suffers from singularities when a minimal state representation is used. This paper presents three alternative implementations of this control approach that eliminate singularities through changes in the control architecture or through redundant formation definitions. These proposed solutions rely on quaternions, dual quaternions, and control implementations that produce singularity-free trajectories while maintaining a cluster level abstraction that allows for simple specification and monitoring. A key component of this work is a novel concept of representing formation shape parameters with dual quaternions. Simulation results show the feasibility of the proposed solutions and illustrate their differences and limitations.

**Keywords** Multirobot · Formation control · Dual quaternions · Mobile robots · UAV · Multicopters

I. Mas (✉)
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) and Instituto Tecnologico de Buenos Aires (ITBA), Buenos Aires, Argentina
e-mail: imas@itba.edu.ar

C. Kitts
Robotic Systems Laboratory, Santa Clara University, Santa Clara CA, USA
e-mail: ckitts@scu.edu

## 1 Introduction

The coordination of groups of mobile robots is a topic of increasing interest in recent years. The ability to distribute sensors, actuators, and task execution promises great improvements in terms of coverage, throughput, and dynamic reconfiguration in the automation of missions. These missions include a slew of possibilities, including search and rescue operations, forest fire control, agricultural monitoring, etc. where the level of automation can range from human-supervised tasks to fully autonomous operation. Such systems have potential to operate in diverse environments, including land, sea, air and space. Examples of specific techniques for mobile robot coordination range from biologically inspired methods [1] to leader-follower architectures [2], virtual structure definitions [3], and potential field configurations [4]. A method of particular interest for multirobot coordination is Cluster Space Control, where the group of

robots is considered as a single entity, namely a cluster, and state variables are defined representing characteristics of the group such as position, orientation and shape. This allows to conceptualize the formation as a full degree of freedom virtual kinematic mechanism that can be intuitively specified, controlled, and monitored by an operator or an autonomous supervisor. The cluster space approach has been successfully implemented in a wide variety of robotic platforms, ranging from land rovers to marine surface vessel systems [5], and for target applications such as escorting/patrolling [6] and multirobot object manipulation [7].

In previous work, implementations of a cluster space (CS) controller relied on the use of an inverse cluster space Jacobian matrix for the case of rate-resolved robots or on cluster space Jacobian transpose matrices for dynamic control architectures [5]. Such configurations allow for a controller to operate directly on the cluster space variables and their errors, but suffer from limitations related to representation singularities. Analogous to singular poses in the classic robot manipulator theory, cluster space singularities occur when the formation attains a pose where some of the formation parameters are undetermined. This also leads to numerical instability in the close vicinity of singular poses, which corrupts control performance. As a distinctive characteristic, CS singularities are often due to representation choices instead of physical constraints as in the case of manipulators. This paper presents alternative implementations of CS control that eliminate singularities through changes in the control architecture or through redundant formation definitions.

Earlier work by the authors have not addressed this topic in detail, as most of the work [5–8] dealt with wheeled ground robots or marine autonomous surface vessels, where the formation operates in a common plane. This characteristic either limits the existence of singular poses or allows then to be easily avoided by choosing alternative definitions to operate around singularity-free poses. One explicit initiative to address this issue was the creation of a state representation management system which actively switched between representations in order to avoid singular configurations [9]. This was done by computing the condition number of the Jacobian matrix for each possible representation and selecting the best-conditioned state definition for use in the control system.

The concept of defining position, orientation and shape characteristics for a formation of mobile robots has been previously proposed by many authors. Initial works relied on leader-follower schemes, using graphs to define a rigid formation shape, and defining trajectories for the leader [10, 11]. Other variations allowed for role changes between leaders and followers [12]. In contrast, CS allows the cluster reference point to be assigned at any location relative to the group of robots; furthermore, it accommodates any fully constraining set of variables to be used to specify formation geometry, and all of these may be arbitrarily varied to allow full degree-of-freedom evolution of the size and shape of the formation. This is contrasted to virtual rigid body techniques [3, 13] and an approach that maps the robot configuration space to a lower dimensional manifold which defines the group's shape as a concentration ellipse [14] where robots are statistically bounded. Because it conserves the dimensionality of the system, the CS approach can be computationally challenging for large numbers of robots and highly coupled cluster state definitions. This can be addressed by using alternate state representations that reduce the number of robots used to define cluster position and/or various shape variables; taken to its limit, the CS approach devolves into a leader-follower chain [10–12], which is completely scalable.

The contributions of this article are three alternate implementations of CS control, all of which are free of representation singularities. The first alternative converts the cluster space specification of motion to the robot space, with a robot space controller implementing these motions in realtime; this changes the nature of group motion and constrains state estimation, but it avoids singular configurations. The other two alternatives avoid minimal state representations through the use of quaternions, a single quaternion representation for formation orientation or dual quaternion representations for describing formation position and shape. In addition to introducing the use of quaternions, controller stability is proven and simulations in the ROS/Gazebo environment verify functionality.

Previous use of quaternions in formation control has simply used quaternions for representing the orientation of each individual robot with respect to a global frame [15] or with respect to one of the other vehicles serving as an orientation reference [16].

The work presented here is the first to use a single quaternion orientation representation of the aggregate formation.

Dual quaternions offer the most compact and computationally efficient screw transformation formalism [17] and can be used as a representation to describe rigid body motions because they simultaneously describe positions and orientations with only eight parameters [18]. Similar to homogeneous transformation matrices, they can describe a complete rigid motion with a single mathematical object. Hence, a sequence of rigid motions is represented by a sequence of dual quaternion multiplications. Dual quaternions are starting to be used in different applications such as control of single free rigid bodies [19], cooperative manipulator arms [18], and leader-follower implementations of multirobot systems [20]. Related work in the literature using dual quaternions is limited to leader-follower configurations [21], where dual quaternions are used to express relative position and orientation of a follower with respect to the leader [22]. This concept has also been extended to leaderless consensus, where the bodies synchronize with each other without the presence of a leader [23]. To the authors' knowledge, the use of dual quaternions has not been extended beyond the representation of position and orientation to concepts such as size and shape. This work presents such an extension in order to represent and control these characteristics in a formation of mobile robots.

## 2 Background

The CS framework for controlling mobile robot formations was formally introduced in [8]. The method relies on the definition of a vector of CS variables that describes the position, orientation, and shape of a virtual kinematic mechanism representing the formation, allowing for full specification and control of the system. For a group of $N$ robots with $m$ degrees of freedom (DOF) each, the framework defines a set of kinematic transforms $\mathbf{c} = \text{KIN}(\mathbf{r})$ –where $\mathbf{r}$ is the vector of $(N \times m)$ robot space variables, (i.e., position and orientation of the robots), and $\mathbf{c}$ is a vector of $(N \times m)$ cluster space variables– as well as a set of inverse kinematic transforms $\mathbf{r} = \text{INVKIN}(\mathbf{c})$. Additionally, relations between velocities in both spaces

can be obtained through the square Jacobian matrix $J(\mathbf{r})$ such that $\dot{\mathbf{c}} = J(\mathbf{r})\,\dot{\mathbf{r}}$.

Throughout this article, a system of two mobile robots with 4 DOF each will be used to demonstrate the techniques being presented. The pose of each robot $i$ is defined by $(x_i, y_i, z_i, \theta_i)$, where $(x_i, y_i, z_i)$ is the robot's position in three translational dimensions and $\theta_i$ is the robot's yaw orientation. Pitch and roll are neglected and not independently specified for these robots, assumptions that are appropriate for the underwater robot and aerial drone multirobot testbeds the authors use for much of their work. It should be noted that, since the use of general 6-DOF robots is not assumed, singularities do not arise when representing the pose of a single robot. This allows to focus the use of quaternions and dual quaternions specifically on the representation of multirobot quantities, and it does not limit the applicability of the proposed approach. Given the use of a formation consisting of two 4-DOF robots, the definition of the cluster is chosen in the following way. The cluster frame is located at the midpoint between robots. The cluster space variables are defined as $\mathbf{c} = (x_c, y_c, z_c, \beta_c, \gamma_c, d_c, \phi_1, \phi_2)$, where $(x_c, y_c, z_c)$ is the cluster position; $\beta_c$ and $\gamma_c$ are the cluster yaw and pitch angles respectively; $d_c$ is the shape parameter that specifies half the distance between robots; and $\phi_i$ is the yaw angle of robot $i$ with respect to the cluster yaw angle, as shown in Fig. 1. The resulting kinematics are therefore:

$$x_c = (x_1 + x_2)/2 \tag{1}$$

$$y_c = (y_1 + y_2)/2 \tag{2}$$

$$z_c = (z_1 + z_2)/2 \tag{3}$$

$$\beta_c = atan2\left(\frac{x_2 - x_1}{y_1 - y_2}\right) \tag{4}$$

$$\gamma_c = atan2\left(\frac{z_1 - z_2}{2\sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}}\right) \tag{5}$$

$$d_c = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}/2 \tag{6}$$

$$\phi_1 = \theta_1 - \beta_c \tag{7}$$

$$\phi_2 = \theta_2 - \beta_c \tag{8}$$

Inverse kinematic relations can also be analytically derived. This approach allows for a control architecture driven by errors calculated in cluster space. Control compensation signals are then transformed through the Cluster Space inverse Jacobian matrix to

**Fig. 1** Cluster space variables definition for a system of two 4-DOF mobile robots
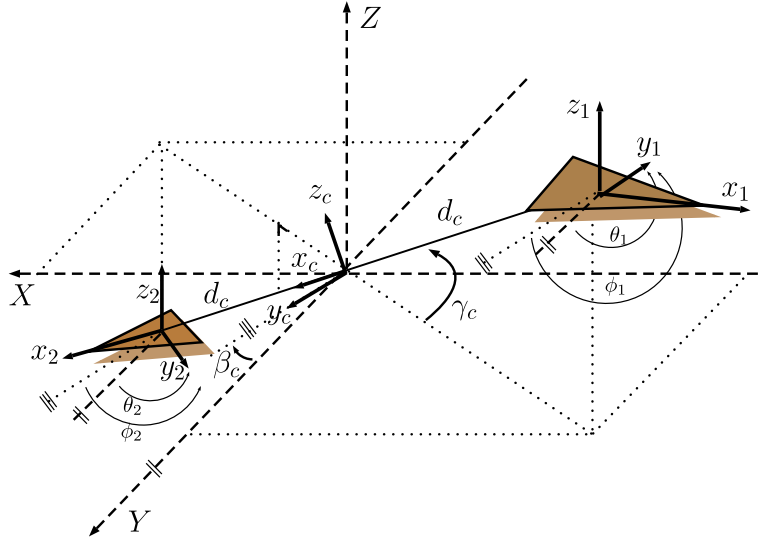
robot level–and ultimately actuator level–commands to produced the required motions. Figure 2 shows the block diagram of a typical CS implementation.

Representation singularities in CS control are situations where certain formation poses result in one or more undetermined values of cluster parameters. Alternatively, singularities can be thought of as poses for which the Jacobian matrix loses rank and is no longer invertible. When the formation reaches a singular pose, the system becomes unstable and its behavior is unpredictable. Of course, this situation must be avoided.

For the two-robot cluster definition presented in Eqs. (1–8), the singular poses are those where $|\gamma_c| = \pi/2$ or where $d_c = 0$. The second case is disregarded given that $d_c = 0$ is also a physical constraint of the system (i.e. two robots being in the same location) and can be neglected for the purpose of this work.

The singular condition for $\gamma_c$ (cluster pitch angle) is critical given that the situation of having one robot

directly above the other may be of interest in particular tasks.

## 3 Singularity-free Approaches

Three control and representation alternatives to produce cluster architectures that do not have singular poses are proposed in this article, allowing for the execution of any trajectory while maintaining the abstraction level that allows for simple specification and monitoring. The first alternative is to maintain the CS framework at the operator interface while implementing the controller in the robot space. The second proposed solution is a unit quaternion representation of the formation orientation, together with a unit quaternion representation of the robot orientations with respect to the cluster, a vector representing the cluster position, and scalars representing the formation size. The third proposed approach is a cluster

**Fig. 2** Cluster Space Control Architecture. Control actions are computed in cluster space and converted to robot space for actuation. Robot sensor information is converted back to cluster space through the forward kinematics to close the loop
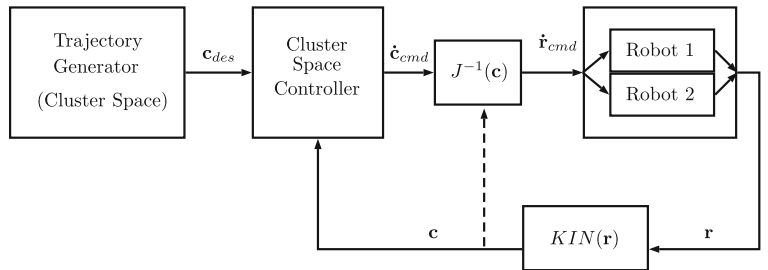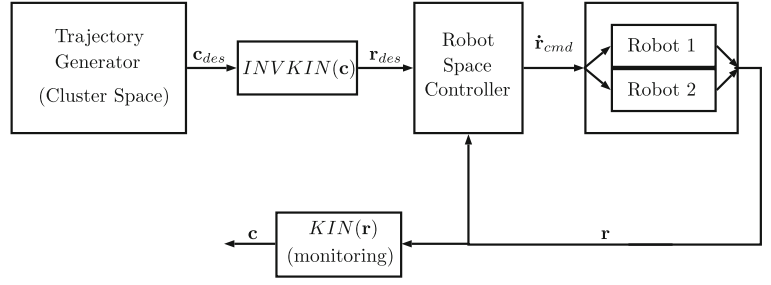
**Fig. 3** Cluster Space
implementation using a
Robot Space controller



definition based on dual unit quaternions, where a compact expression can be used to represent position, orientation, and shape of the formation.

### 3.1 Robot-Space Formation Control

In this architecture, the desired formation trajectories are specified for all CS variables as before, but the inverse kinematic transform is used to obtain the corresponding desired trajectory in robot space. Errors between these trajectories and the measurements from the robot sensors are used to operate a robot-space controller that commands the vehicles. In this implementation, the inverse Jacobian matrix is not used to transform controller commands and singularities do not arise. Figure 3 shows a block diagram of this implementation.

This singularity-free architecture has an important limitation: as the controller operates in robot space,

motions evolve in robot space and are not 'well behaved' as seen from the formation perspective. This is demonstrated in Fig. 4 for the example of a 180° step input command for the formation yaw angle. A conventional CS controller produces the motion shown in Fig. 4a, moving the robots through the shortest CS path, producing what from the robot space perspective looks like a circular motion, and therefore keeping the formation shape as they move. On the other hand, a controller in robot space commands the robots through the shortest robot-space path, producing the motions shown in Fig. 4b. In this case, the robots simply swap positions and the formation shape is not naturally maintained over the motion.

### 3.2 Quaternion-based Formation Control

Quaternions can be considered as extensions of complex numbers to $\mathbf{R}^4$. A unit quaternion can be used



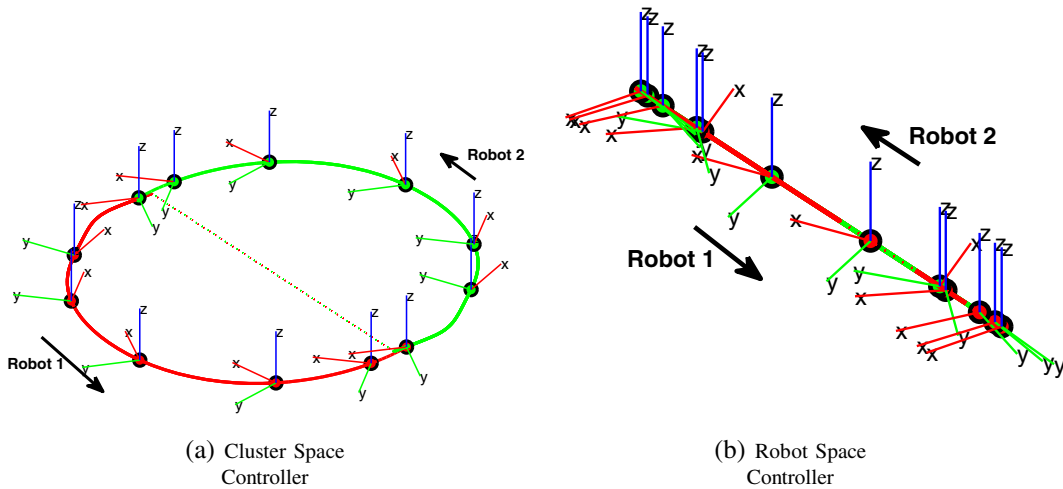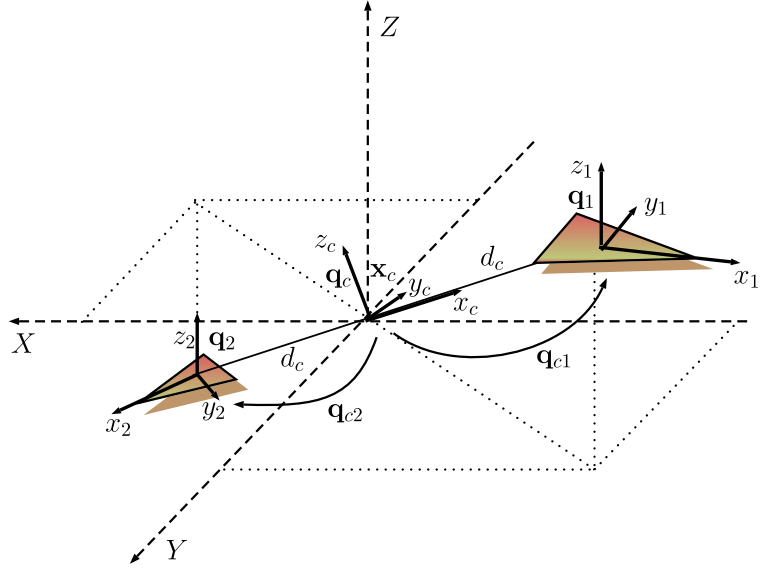(a) Cluster Space
Controller

(b) Robot Space
Controller

**Fig. 4** Controllers operating in different spaces result in different formation behaviors for a step trajectory with similar initial and final positions

**Fig. 5** Quaternion-based cluster frame definitions

to describe a rotation. For a frame rotation about a unit axis $\mathbf{n}$ with angle $0 \leq \theta < 2\pi$, there is a unit quaternion $\mathbf{q} = [cos(\theta/2); sin(\theta/2)\mathbf{n}]$ representing such rotation. Quaternion representations have been widely used to represent and control the relative attitude of one vehicle with respect to another vehicle, as in the case of leader-follower configurations [24], but they have not been used to represent the orientation of a formation as a whole. A formation of robots can be represented by a cluster position vector and a cluster orientation unit quaternion. Additionally, scalars describing the formation shape and quaternions describing the orientation of the robots with respect to the cluster can be defined.

For the 4-DOF two-robot system presented in this article, the cluster space can be defined as $\mathbf{c} = (\mathbf{x}_c, \mathbf{q}_c, d_c, \mathbf{q}_{c1}, \mathbf{q}_{c2})$, where $\mathbf{x}_c = (x_c, y_c, z_c)$ is the centroid of the two robots, $\mathbf{q}_c$ is a unit quaternion representing a frame aligned with a segment going from robot 2 to robot 1, $d_c$ is half the length of such segment, and $\mathbf{q}_{ci}$ represents the orientation of robot $i$ with respect to the cluster orientation. It should be noted that quaternions naturally represent 3-DOF rotations, therefore a system of two full 6-DOF robots would be represented in the same way. The forward kinematic transforms are

$$\mathbf{x}_c = (\mathbf{x}_1 + \mathbf{x}_2)/2 \tag{9}$$

$$\mathbf{q}_c = [cos(\eta/2); sin(\eta/2)\mathbf{n}_{12}] \tag{10}$$

$$d_c = \|\mathbf{x}_1 - \mathbf{x}_2\|/2 \tag{11}$$

$$\mathbf{q}_{c1} = \mathbf{q}_c^* \circ \mathbf{q}_1 \tag{12}$$

$$\mathbf{q}_{c2} = \mathbf{q}_c^* \circ \mathbf{q}_2, \tag{13}$$

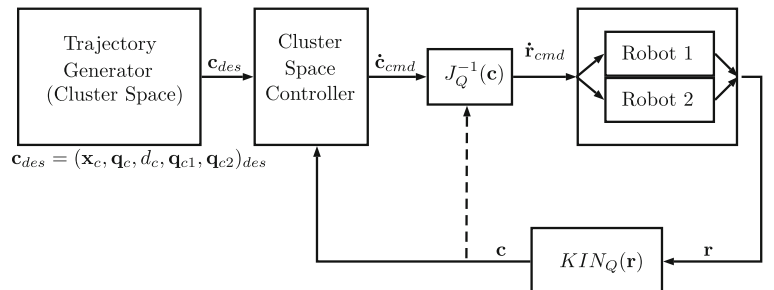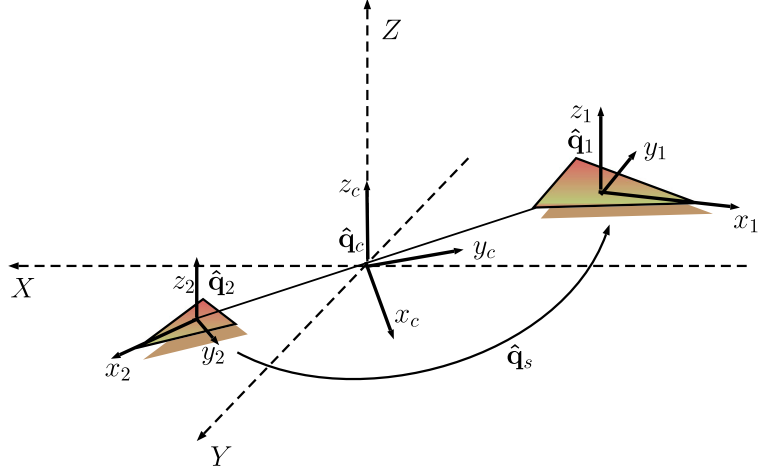**Fig. 6** Cluster Space implementation with quaternion definition

**Fig. 7** Representation of cluster and robots dual quaternion frames



with

$$\eta = atan2\left(\frac{\|\hat{i} \times (\mathbf{x}_1 - \mathbf{x}_2)\|}{\hat{i} \cdot (\mathbf{x}_1 - \mathbf{x}_2)}\right) \qquad (14)$$

$$\mathbf{n}_{12} = \frac{\hat{i} \times (\mathbf{x}_1 - \mathbf{x}_2)}{\|\hat{i} \times (\mathbf{x}_1 - \mathbf{x}_2)\|}, \qquad (15)$$

where $\hat{i} = (1, 0, 0)$, and the operator $\circ$ indicates quaternion multiplication. Figure 5 shows the cluster and robot frames for the chosen definition. The inverse kinematics are

$$\mathbf{x}_1 = \mathbf{x}_c + \mathbf{q}_c \circ \mathbf{d}_c \circ \mathbf{q}_c^* \qquad (16)$$

$$\mathbf{q}_1 = \mathbf{q}_c \circ \mathbf{q}_{c1} \qquad (17)$$

$$\mathbf{x}_2 = \mathbf{x}_c - \mathbf{q}_c \circ \mathbf{d}_c \circ \mathbf{q}_c^* \qquad (18)$$

$$\mathbf{q}_2 = \mathbf{q}_c \circ \mathbf{q}_{c2}, \qquad (19)$$

where $\mathbf{d}_c = (0, d_c, 0, 0)$. Motions are specified in cluster space and desired formation trajectories are compared to the formation state recovered from robot sensor information through the forward kinematic transform (9–13) in a controller that operates in the space of the cluster. The control compensation is converted to robot space commands using the inverse Jacobian matrix derived from Eqs. (16–19). Although the inverse Jacobian matrix is not square as it was in the original Cluster Space formulation, it has full rank equal to the total DOF of the physical system for any attainable formation pose, therefore singularities do not arise. The controller implements proportional compensation for the position and shape variables, $\dot{\mathbf{x}}_{c\,cmd} = -K_x \mathbf{e}_{xc} + \dot{x}_{c\,des}$, where $\mathbf{e}_{xc} = \mathbf{x}_c - \mathbf{x}_{c\,des}$ and $\dot{d}_{c\,cmd} = -k_d e_{dc} + \dot{d}_{c\,des}$, where $e_{dc} = d_c - d_{c\,des}$. To control orientations, quaternion errors $\mathbf{qe}_j = \mathbf{q}_j \circ \mathbf{q}_{j\,des} = (q_{0j}; \tilde{\mathbf{q}}_j); j = \{c, c_1, c_2\}$ are minimized using:

$$\dot{\mathbf{q}}_{j\,cmd} = \frac{1}{2}\mathbf{q}_j \circ \begin{bmatrix} 0 \\ -\text{sgn}(q_{0j})K_w \tilde{\mathbf{q}}_j \end{bmatrix} \qquad (20)$$

where $K_w \in \mathbf{R}^{3\times3}$ is a diagonal matrix of positive constant gains, and $\text{sgn}(q_{0j})$ is the signum function of the real part of the error quaternion. Figure 6 shows the described implementation.

**Fig. 8** Implementation of dual quaternion cluster space
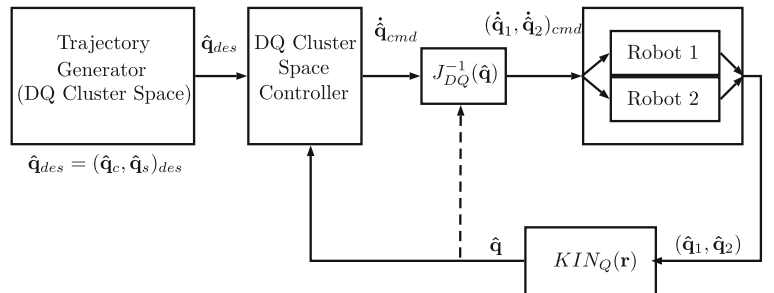
*Proof* Proposing the positive-definite Lyapunov candidate function in terms of the errors in cluster space:

$$V(\mathbf{c}) = \frac{1}{2}\left(\mathbf{e}_{xc}^* \mathbf{e}_{xc} + e_{dc}^2 + \sum_{j=\{c,c_1,c_2\}} (1 - q_{0j}^2 + \tilde{\mathbf{q}}_j^* \tilde{\mathbf{q}}_j)\right),$$
(21)

the time derivative of the Lyapunov function is

$$\dot{V}(\mathbf{c}) = \mathbf{e}_{xc}^* \dot{\mathbf{e}}_{xc} + e_{dc}\dot{e}_{dc} + \sum_j \left(-q_{0j}\dot{q}_{0j} + \tilde{\mathbf{q}}_j^* \dot{\tilde{\mathbf{q}}}_j\right) \quad (22)$$

Using the proposed controller and relying on the fact that $\dot{\mathbf{q}}\mathbf{e}_j = (\dot{q}_{0j}; \dot{\tilde{\mathbf{q}}}_j) = \frac{1}{2}\mathbf{q}\mathbf{e}_j\,\omega_{j\,cmd}$, where

$\omega_{j\,cmd} = -\mathrm{sgn}(q_{0j})K_w\tilde{\mathbf{q}}_j$, or expressed as a matrix multiplication:

$$\dot{\mathbf{q}}\mathbf{e}_j = \frac{1}{2}\begin{bmatrix} -\tilde{\mathbf{q}}_j^* \\ q_{0j}I_3 + \tilde{\mathbf{Q}}_{xj} \end{bmatrix}\omega_{j\,cmd} \quad (23)$$

where $\tilde{\mathbf{Q}}_{xj}$ is the cross-product operator skew-symmetric matrix form of $\tilde{\mathbf{q}}_j$ and $I_3$ is the 3 by 3 identity matrix. Replacing (23) in (22):

$$\dot{V}(\mathbf{c}) = \mathbf{e}_{xc}^* K_x \mathbf{e}_{xc} + k_d e_{dc}^2 + \ldots$$

$$\sum_j \Big( -\frac{1}{2}q_{0j}\tilde{\mathbf{q}}_j^* K_w \mathrm{sgn}(q_{0j})\tilde{\mathbf{q}}_j \ldots$$

$$-\frac{1}{2}\tilde{\mathbf{q}}_j^* q_{0j}K_w \mathrm{sgn}(q_{0j})\tilde{\mathbf{q}}_j \ldots$$

$$-\frac{1}{2}\tilde{\mathbf{q}}_j^* \tilde{\mathbf{Q}}_{xj}K_w \mathrm{sgn}(q_{0j})\tilde{\mathbf{q}}_j\Big) \quad (24)$$

**Fig. 9** Baseline cluster space trajectory tracking results. Singularities become apparent in $\beta_c$ at $t = 27.5s$
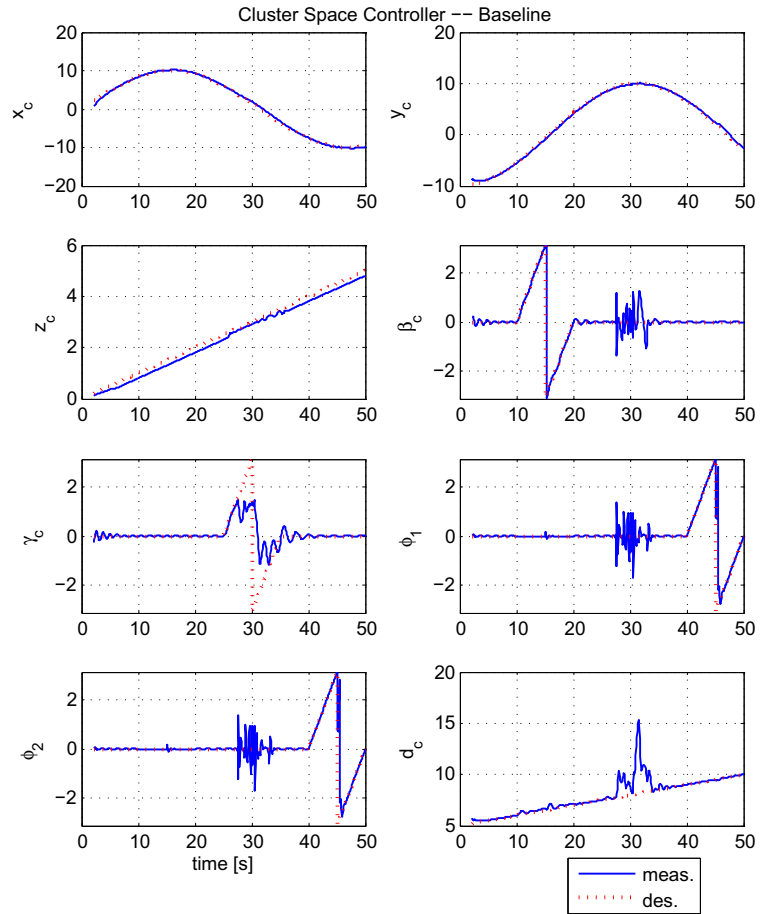
Noting that $K_w$ is a diagonal positive matrix, and when multiplied by the skew-symmetric matrix $\tilde{\mathbf{Q}}_{xj}$ the result is skew-symmetric, then the last term of (24) has the form $X^*SX = 0$, where $S$ is skew-symmetric. Therefore, (24) simplifies to

$$\dot{V}(\mathbf{c}) = -\,\mathbf{e}_{xc}^* K_x \mathbf{e}_{xc} - k_d e_{dc}^2 \cdots$$
$$-\sum_{j=\{c,c_1,c_2\}} \left(\tilde{\mathbf{q}}_j^* | q_{0j} | K_w \tilde{\mathbf{q}}_j\right) \leq 0 \qquad (25)$$

rendering the system stable. $\qquad\qquad\qquad\square$

This definition is singularity-free due to the redundant orientation representation provided by unit quaternions, and maintains the desired level of control abstraction. The drawback of this approach is the increase in the number of cluster space variables to be tracked; in the case of two 4-DOF robots, the resulting 8-DOF system is represented by 16 variables, adding to the computational complexity.

## 3.3 Dual Quaternion Formation Control

A dual number is defined as $\hat{\alpha} = a + \epsilon b$ with $\epsilon^2 = 0$, but $\epsilon \neq 0$, where $a$ and $b$ are real numbers, called the principal part and the dual part, respectively. A dual quaternion can be treated as a dual number with quaternion components, i.e., $\hat{\mathbf{q}} = \mathbf{q}_r + \epsilon\mathbf{q}_d$, where $\mathbf{q}_r$ and $\mathbf{q}_d$ are quaternions.

The main contribution of this section is the representation of position, orientation and shape concepts by dual quaternions. The dual quaternion $\hat{\mathbf{q}} = \mathbf{q} + \epsilon\frac{1}{2}\mathbf{p} \circ \mathbf{q}$ represents a rigid motion where the principal part $\mathcal{P}(\hat{\mathbf{q}}) = \mathbf{q}$ represents the rotation and the dual part $\mathcal{D}(\hat{\mathbf{q}}) = \frac{1}{2}\mathbf{p} \circ \mathbf{q}$ indirectly represents the translation. The translation can be retrieved using dual quaternion operation $\mathbf{p} = 2\mathcal{D}(\hat{\mathbf{q}}) \circ \mathcal{P}(\hat{\mathbf{q}})^T$, where $\mathbf{p} = (0, x, y, z)^T$.

The aim of this section is to represent the formation parameters with a set of dual quaternions.



Fig. 10 Tracking results using the robot space controller
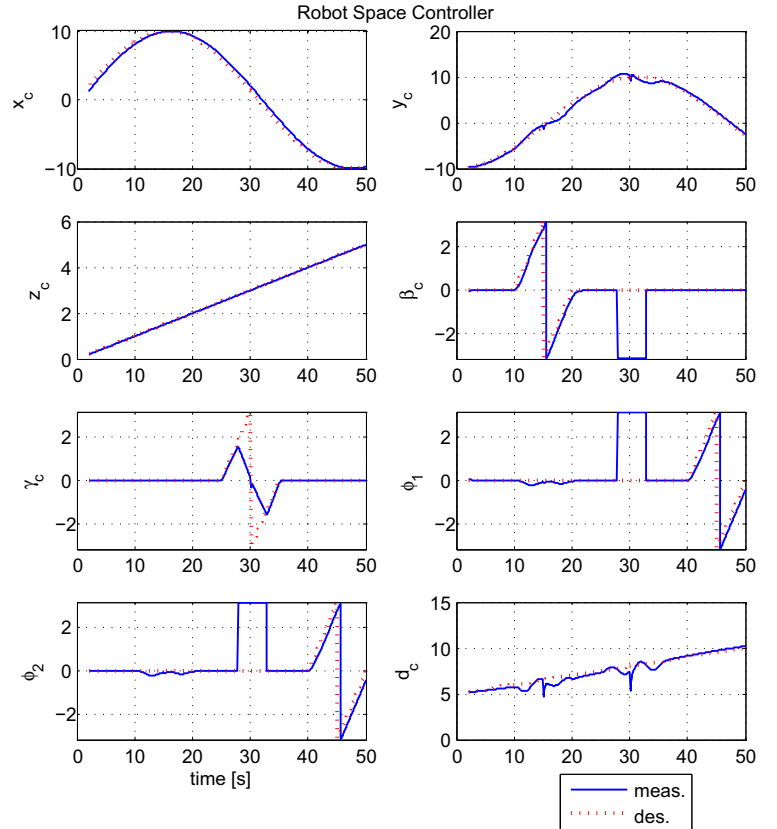
In particular, the goal is to integrate in a compact and redundant –i.e. singularity free– representation the orientation of the formation and its position and shape using respectively the principal part and dual part of dual quaternions.

Using the presented two-robot system, a cluster definition based on two dual quaternions is proposed, where one quaternion is related to the center of the formation and the other carries information of its shape. This concept is based on developments for dual quaternion-defined multi-arm manipulation [25]. Given a dual quaternion representation of the robots' position and orientation: $\hat{\mathbf{q}}_1 = \mathbf{q}_1 + \epsilon \frac{1}{2} \mathbf{p}_1 \circ \mathbf{q}_1$ and $\hat{\mathbf{q}}_2 = \mathbf{q}_2 + \epsilon \frac{1}{2} \mathbf{p}_2 \circ \mathbf{q}_2$, where $\mathbf{q}_i$ is the quaternion representing the orientation of robot $i$ and $\mathbf{p}_i = (0, x_i, y_i, z_i)$

represents its position, then the cluster can be specified with a set of two dual quaternions $\hat{\mathbf{q}}_c$, $\hat{\mathbf{q}}_s$, where

$$\hat{\mathbf{q}}_c = \hat{\mathbf{q}}_2 \circ (\hat{\mathbf{q}}_2^* \circ \hat{\mathbf{q}}_1)^{1/2} \tag{26}$$
$$\hat{\mathbf{q}}_s = \hat{\mathbf{q}}_2^* \circ \hat{\mathbf{q}}_1. \tag{27}$$

Figure 7 shows the physical interpretation of the cluster dual quaternions of Eqs. (26, 27), where $\hat{\mathbf{q}}_c$ represents an average frame located between the two robot frames, and $\hat{\mathbf{q}}_s$ indicates the shape with a representation of the relative rotation and separation between robot frames. Furthermore, the inverse kinematics are defined by $\hat{\mathbf{q}}_1 = \hat{\mathbf{q}}_c \circ (\hat{\mathbf{q}}_s^{*1/2})^*$ and $\hat{\mathbf{q}}_2 = \hat{\mathbf{q}}_c \circ (\hat{\mathbf{q}}_s^{1/2})^*$. Given these definitions, a dual quaternion



**Fig. 11** Parameters tracking results using the quaternion definition
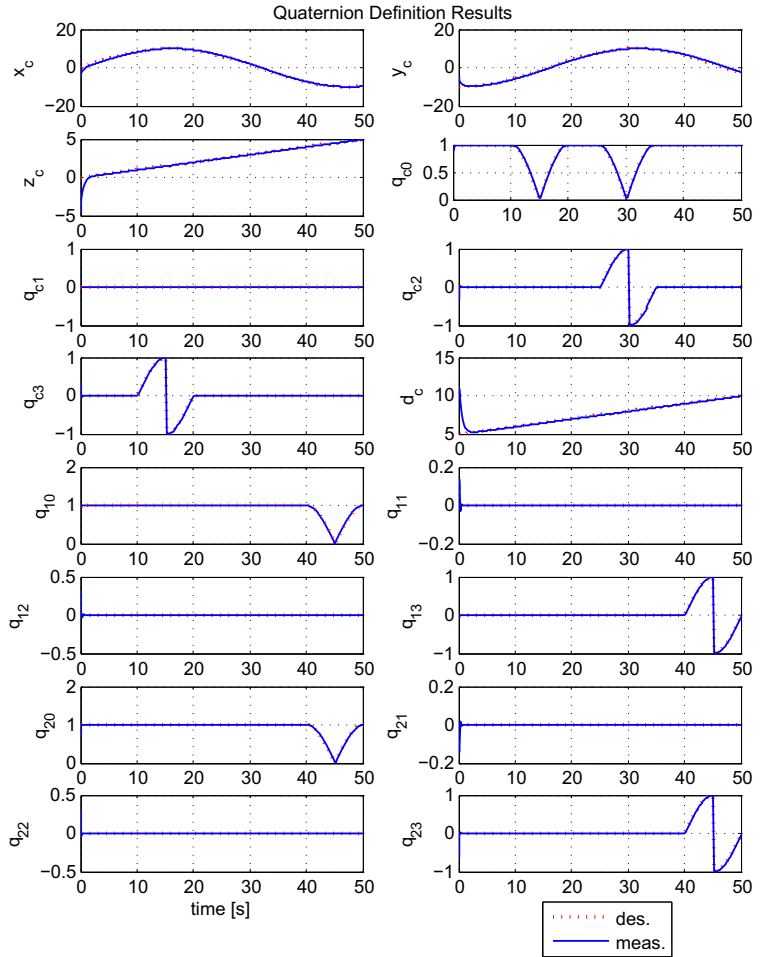
cluster Jacobian matrix can be derived and used in a cluster dual quaternion space controller as shown in Fig. 8.

The proposed controller considers the dual quaternion errors $\hat{\boldsymbol{q}}e_k = \hat{\boldsymbol{q}}_k \circ \hat{\boldsymbol{q}}_{k\,des} = (q_{p0_k}; \tilde{\hat{\mathbf{q}}}_{p_k}; 0; \tilde{\hat{\mathbf{q}}}_{d_k}); k = \{c, s\}$, such that:

$$\dot{\hat{\mathbf{q}}}_{k\,cmd} = \frac{1}{2}\hat{\mathbf{q}}_k \circ \begin{bmatrix} 0 \\ -\text{sgn}(q_{p0_k})K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k} \\ 0 \\ -\text{sgn}(q_{p0_k})K_{dw}\tilde{\hat{\mathbf{q}}}_{d_k} \end{bmatrix} \quad (28)$$

where $K_{dw} \in \mathbf{R}^{3\times3}$ is a diagonal matrix of positive constant gains, and $\text{sgn}(q_{p0_k})$ is the signum function of the real element of the principal part of the dual quaternion error.

*Proof* Proposing the Lyapunov candidate function in cluster space

$$V(\mathbf{c}) = \sum_{k=\{c,s\}} \left(1 - q_{p0k}^2 + \tilde{\hat{\mathbf{q}}}_{p_k}^* \tilde{\hat{\mathbf{q}}}_{p_k} + \tilde{\hat{\mathbf{q}}}_{d_k}^* \tilde{\hat{\mathbf{q}}}_{d_k}\right), \quad (29)$$

its time derivative has the form

$$\dot{V}(\mathbf{c}) = \sum_{k=\{c,s\}} \left(-2q_{p0k}\dot{q}_{p0k} + 2\tilde{\hat{\mathbf{q}}}_{p_k}^* \dot{\tilde{\hat{\mathbf{q}}}}_{p_k} + 2\tilde{\hat{\mathbf{q}}}_{d_k}^* \dot{\tilde{\hat{\mathbf{q}}}}_{d_k}\right). \quad (30)$$

The time derivative of the dual quaternion error $\dot{\hat{\boldsymbol{q}}}e_k = (\dot{q}_{p0_k}; \dot{\tilde{\hat{\mathbf{q}}}}_{p_k}; 0; \dot{\tilde{\hat{\mathbf{q}}}}_{d_k}); k = \{c, s\}$ can be expressed as

$$\dot{\hat{\boldsymbol{q}}}e_k = \frac{1}{2}\begin{bmatrix} -\tilde{\hat{\mathbf{q}}}_{p_k}^* & 0 \\ q_{p0k}I_3 + \tilde{\mathbf{Q}}_{xpk} & 0 \\ -\tilde{\hat{\mathbf{q}}}_{d_k}^* & -\tilde{\hat{\mathbf{q}}}_{p_k}^* \\ \tilde{\mathbf{Q}}_{xdk} & q_{p0k}I_3 + \tilde{\mathbf{Q}}_{xpk} \end{bmatrix}\begin{bmatrix} \omega_{pk\,cmd} \\ \omega_{dk\,cmd} \end{bmatrix} \quad (31)$$

where $\tilde{\mathbf{Q}}_{xpk}$ and $\tilde{\mathbf{Q}}_{xdk}$ are the cross-product operator skew-symmetric matrix form of $\tilde{\hat{\mathbf{q}}}_{p_k}$ and $\tilde{\hat{\mathbf{q}}}_{d_k}$ respectively, and $I_3$ is the $(3 \times 3)$ identity matrix. Additionally, $\omega_{pk\,cmd}$ and $\omega_{dk\,cmd}$ are defined by the controller as $\omega_{pk\,cmd} = -\text{sgn}(q_{p0_k})K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k}$ and $\omega_{dk\,cmd} =$

$-\text{sgn}(q_{p0_k})K_{dw}\tilde{\hat{\mathbf{q}}}_{d_k}$. These expressions can be substituted in (30) to obtain:

$$\dot{V}(\mathbf{c}) = \sum_{k=\{c,s\}} \big( -\tilde{\hat{\mathbf{q}}}_{p_k}^* |q_{p0k}|K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k} \ldots$$
$$- \tilde{\hat{\mathbf{q}}}_{p_k}^* |q_{p0k}|K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k} \ldots$$
$$- \tilde{\hat{\mathbf{q}}}_{p_k}^* \text{sgn}(q_{p0k})\tilde{\mathbf{Q}}_{xpk}K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k} \ldots$$
$$- \tilde{\hat{\mathbf{q}}}_{d_k}^* \tilde{\mathbf{Q}}_{xdk}\text{sgn}(q_{p0k})K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k} \ldots$$
$$- \tilde{\hat{\mathbf{q}}}_{d_k}^* |q_{p0k}|K_{dw}\tilde{\hat{\mathbf{q}}}_{d_k} \ldots$$
$$- \tilde{\hat{\mathbf{q}}}_{d_k}^* \tilde{\mathbf{Q}}_{xpk}\text{sgn}(q_{p0k})K_{dw}\tilde{\hat{\mathbf{q}}}_{d_k}\big). \quad (32)$$

Noting that $K_{dw}$ is a diagonal positive matrix, and when multiplied by the skew-symmetric matrix $\tilde{\mathbf{Q}}_{xpk}$ the result is skew-symmetric, then the third term of (32) has the form $X^*SX = 0$, where $S$ is skew-symmetric and that the fourth and sixth terms are equal and opposite and cancel each other, (32) simplifies to

$$\dot{V}(\mathbf{c}) = \sum_{k=\{c,s\}} \left(-2\tilde{\hat{\mathbf{q}}}_{p_k}^* |q_{p0k}|K_{dw}\tilde{\hat{\mathbf{q}}}_{p_k} - \tilde{\hat{\mathbf{q}}}_{d_k}^* |q_{p0k}|K_{dw}\tilde{\hat{\mathbf{q}}}_{d_k}\right) \leq 0 \quad (33)$$

rendering the system stable. □

Similar to the quaternion definition in the previous section, dual quaternions add complexity to the system by resulting in a 16-variable description. Additionally, specification and monitoring is less transparent due to the increased abstraction of the parameterization. Nonetheless, an additional conversion of variables to an operator-friendly representation for monitoring is possible. On the positive side, the compact definition makes for an elegant solution and the controller operating in the dual quaternion formation space produces well-behaved motions as described in Section 3.1.

## 4 Results

Simulations in the MATLAB environment using a common trajectory are used to illustrate the functionality of each proposed architecture and to compare the behavior in each case. Then, a high-fidelity simulation of a formation of commercially-available IRIS quadrotors from 3D Robotics in the ROS/Gazebo environment is used to obtain results in a more realistic scenario.

Dual Quaternion Definition Results
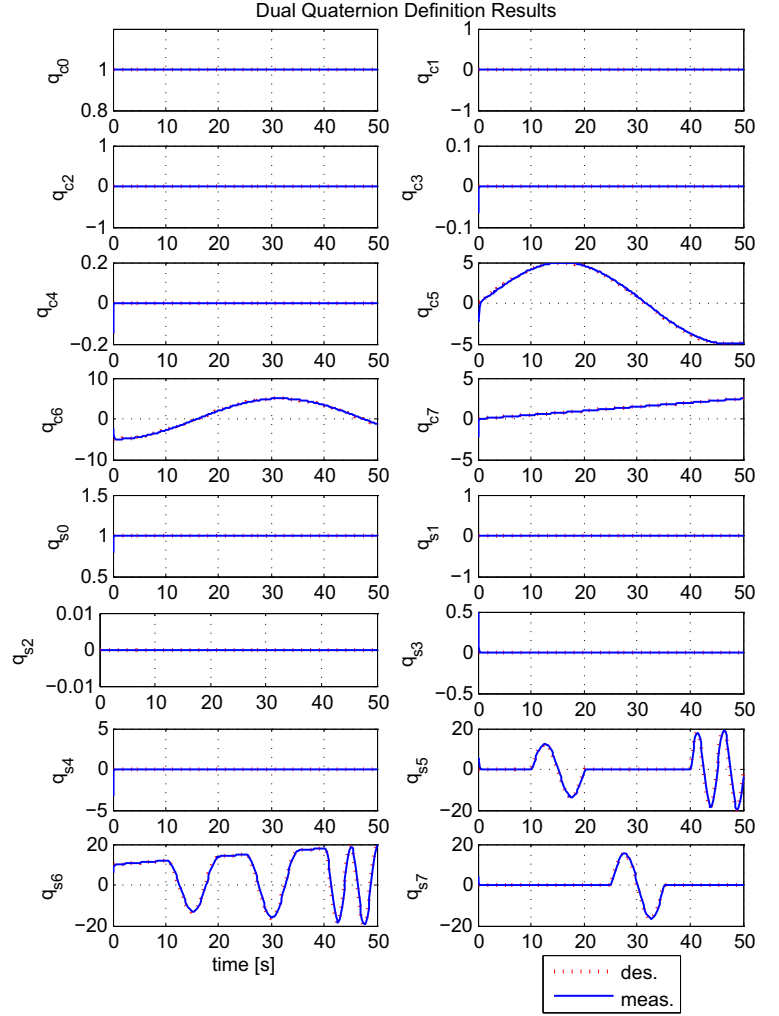
## 4.1 Simulations in the MATLAB Environment

The commanded trajectory is a rising spiral, with the formation's centroid following a circle in the $xy$ plane and a linear trajectory in $z$. At $t = 10s$, the formation performs a $2\pi$ rotation in yaw, at $t = 25s$, a $2\pi$ rotation in pitch, and at $t = 40s$, the $\phi_1$ and $\phi_2$ angles rotate $2\pi$. Simultaneously, the formation follows a linear variation in the inter-robot distance $d_c$. Of particular interest is the value of $\gamma_c$ (formation pitch) in the range $[0, 2\pi]$, where the existence of singularities becomes apparent.

First, the standard cluster space controller is used to track the proposed trajectory. The results are shown in Fig. 9. When the $\gamma_c$ reference signal reaches $\pi/2$, the system hits a singular pose and is unable to track

**Table 1** Controller
parameter values for test
cases of Section 4.1

| Method | Gains |
|---|---|
| Cluster Space | $K_{CS} = diag(1; 1; 1; 10; 10; 5; 5; 10)$ |
| Robot Space | $K_{RS} = diag(1.5; 1.5; 1.5; 6; 6; 1.5; 1.5; 1.5)$ |
| Quat. | $K_x = diag(2; 2; 2), k_d = 20, K_w = diag(20; 20; 20)$ |
| Dual Quat. | $K_{dw} = diag(15; 15; 15)$ |

**Fig. 13** Error of formation parameters for the test trajectory using the different proposed methods
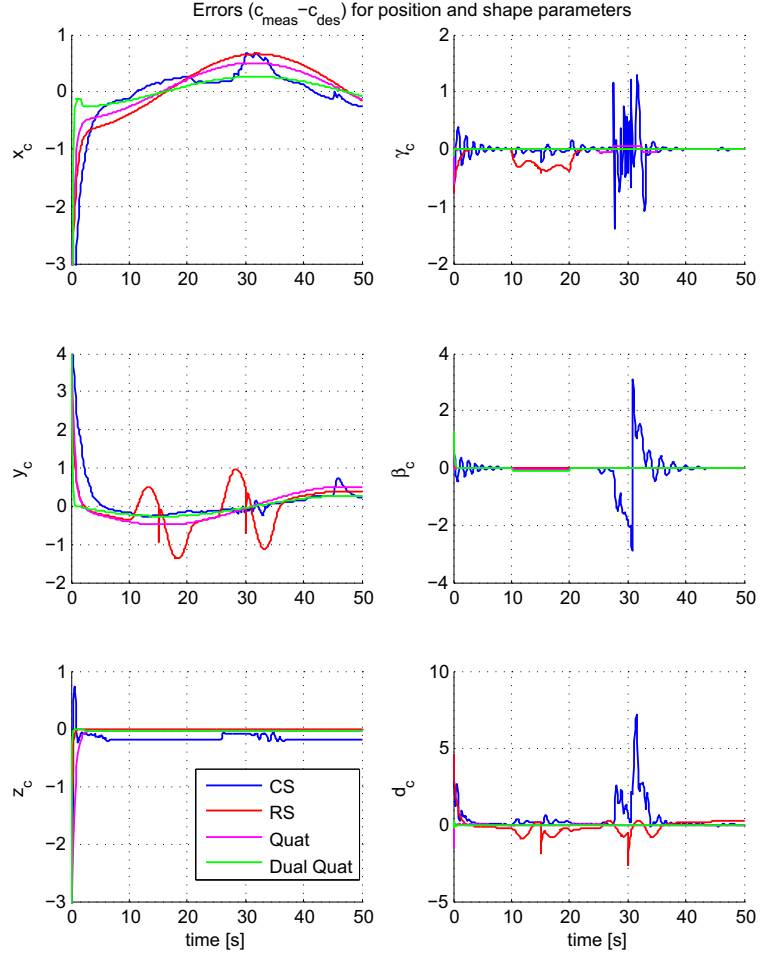
the reference any further. At this point, the state of $\beta_c$ is undetermined and therefore can not be controlled, rendering the system unstable. Furthermore, given Eqs. (7, 8) the same behavior is observed for the $\phi_i$ parameters.

Figure 10 shows the cluster space trajectories as tracked by the robot space controller presented in Section 3.1. The tracking is stable, and there are no singular poses, but the cluster variables fail to recover the true state of the system due to representation ambiguities. In particular, when $|\gamma_c| > \pi/2$, the kinematic transforms interpret this pose as $\beta_c = \pm\pi$ and $|\gamma_c| < \pi/2$. A similar effect is seen in the $\phi_i$ parameters. This discrepancy only affects the monitoring task as the robot-space controller is relying on actual robot poses, and the trajectories can be tracked successfully at all times.

Next, the quaternion definition of Section 3.2 is used to track the proposed trajectory. The references are previously converted to the equivalent
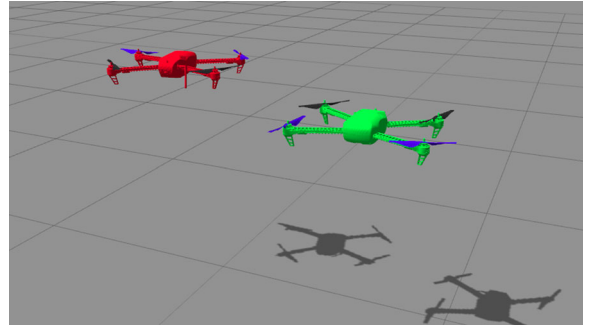


**Fig. 14** ROS/Gazebo Simulator graphical interface showing two 3D Robotics Iris multicopters

| Method | Gains |
|---|---|
| Cluster Space | $K_{CS} = diag(0.06; 0.06; 0.03; 0.02; 0.02; 0.03; 0.03; 0.03)$ |
| Robot Space | $K_{RS} = diag(0.06; 0.06; 0.03; 0.02; 0.02; 0.03; 0.03; 0.03)$ |
| Quat. | $K_x = diag(0.7; 0.7; 0.7)$, $k_d = 0.4$, $K_w = diag(1; 1; 1)$ |
| Dual Quat. | $K_{dw} = diag(0.5; 0.5; 0.5)$ |

quaternion reference trajectories. Figure 11 shows reference and measured signals for each of the 16 variables as defined in Eqs. (9–13). The robots successfully track the desired trajectories.
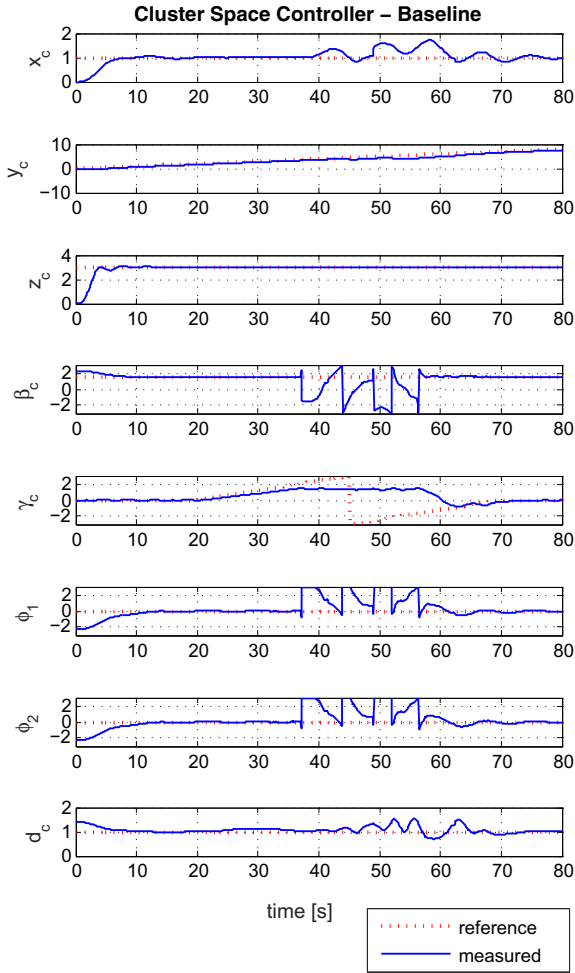
Finally, the trajectory is tracked by a system implementing the dual quaternion cluster definition of Section 3.3. Again, appropriate trajectories are generated to produce the same experiment. Results for each of the 16 variables are shown in Fig. 12, where all trajectories are tracked and no singularities are present.

The simulations consider 4-DOF holonomic kinematic models of the robots. The parameters of the
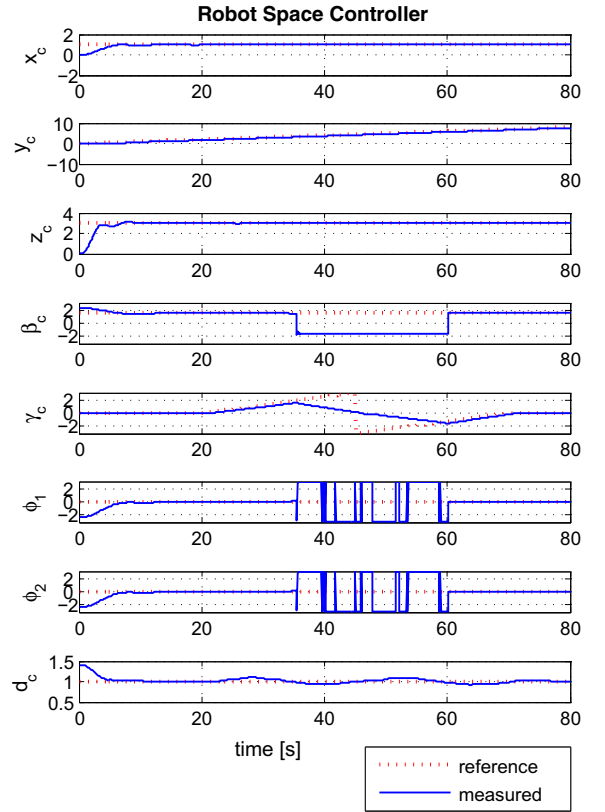


**Fig. 15** Trajectory tracking results using the cluster space baseline definition in the ROS/Gazebo Simulator



**Fig. 16** Trajectory tracking results using the robot space definition in the ROS/Gazebo Simulator

controllers implemented in the simulations are shown in Table 1. All controller are proportional and the gains were empirically tuned.

In order to compare these results, tracking errors are presented in Fig. 13. To make the comparison useful, variables in the different spaces were transformed to the equivalent values in CS, and errors ($e_c = c_{meas} - c_{des}$) are shown for the position and shape parameters. All values converge to zero with a steady state error corresponding to the lack of integral compensation of the controllers. Moreover, a significant error can be seen in the RS controller as the formation performs the pitch and yaw maneuvers. Such errors could be reduced with appropriate further tunning of the controller. The error in the CS implementation described in Section 2 is readily noticeable, where at around $t = 25s$, the pitch maneuver hits a singular pose, and the error grows significantly as the controller cannot track the reference.
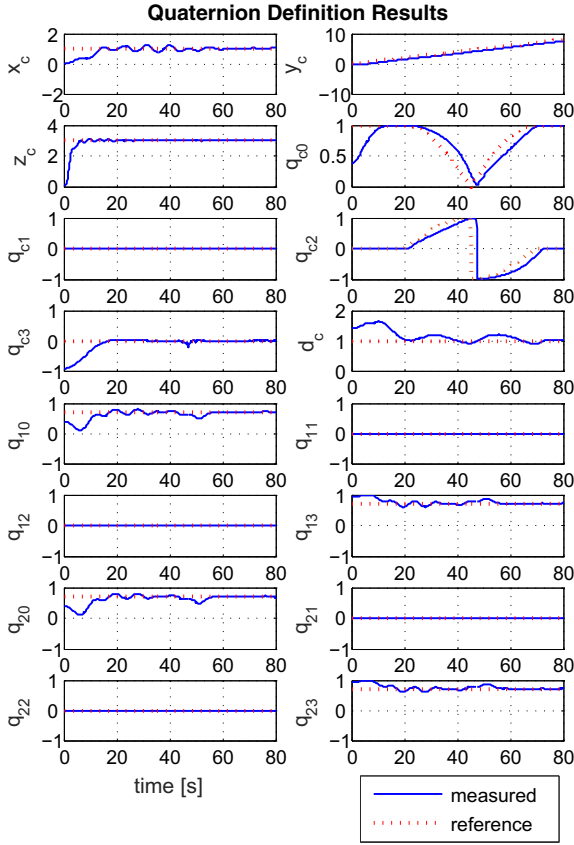
## 4.2 Simulations in the ROS/Gazebo Environment

To demonstrate functionality in a more realistic environment, a simulator in the Robot Operating System (ROS) and Gazebo environment was used. ROS is an open-source *de facto* standard in robotics research and facilitates fast development and testing [26] of robotic systems. Two IRIS multicopters from 3D-Robotics were modeled with a Gazebo Simulator plug-in developed by the Autonomous System Lab of ETH Zurich University (Fig. 14). Each multicopter runs a Linux port of the PX4 autopilot firmware [27] with a MAVROS interface. This architecture allows a seamless transition of the proposed algorithms from the simulator to the actual hardware.

The trajectory presented is a linear motion in the $y$ direction at constant altitude and a robot separation of $1m$, with a $2\pi$ rotation of the formation around the $y$ axis between times $t = 20s$ and $t = 60s$,



**Fig. 17** Trajectory tracking results using the quaternion definition in the ROS/Gazebo Simulator
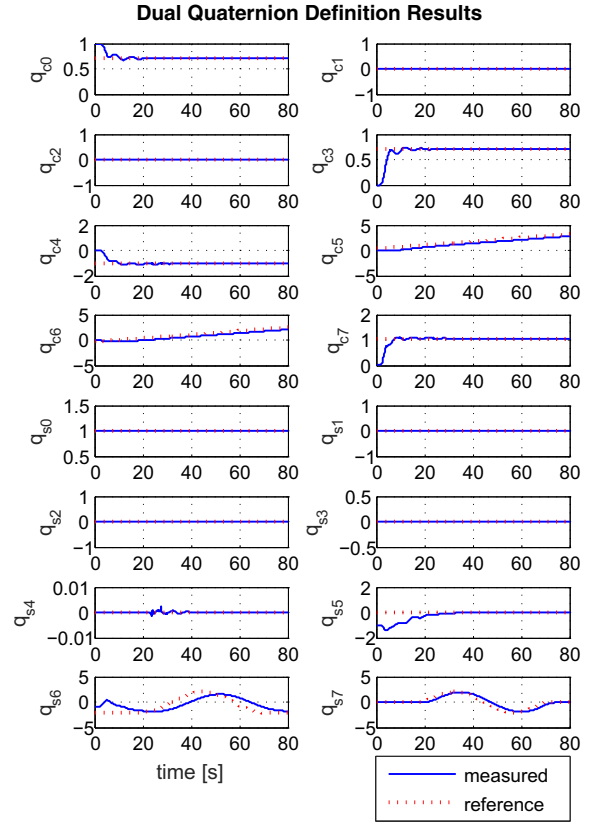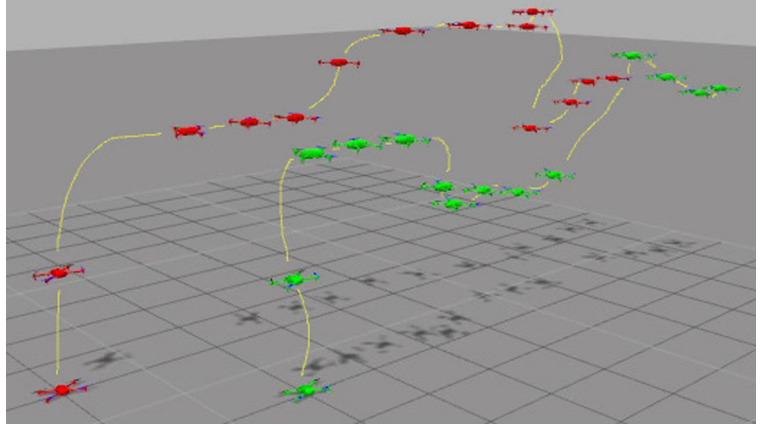


**Fig. 18** Trajectory tracking results using the dual quaternion definition in the ROS/Gazebo Simulator

**Fig. 19** Time lapse of the
controller implementing the
cluster space baseline
definition in the
ROS/Gazebo Simulator



producing a helical motion. Controllers with gains
shown in Table 2 for each proposed formation con-
trol method are implemented and results are shown
in Figs. (15–18). Similarly to the previous results of
Section 4.1, the cluster space baseline fails to fol-
low the reference trajectory when the singularity is
reached at around $t = 30s$ (Fig. 15). The robot-space
controller of Fig. 16 shows that the trajectory is cor-
rectly followed, but the method fails to recover the
real state of the formation from the measurements due
to representation ambiguities. Finally, the quaternion
and dual quaternion representations (Figs. 17 and 18)
are able to follow the references correctly and the for-
mation state is adequately recovered from the sensor
information.

Figure 19 shows a time-lapse of the simulator
graphical interface for the cluster space baseline con-
troller where the helical motion cannot be achieved as
the cluster hits the singularity at $\gamma_c = \pi/2$. Figure 20
shows the resulting trajectory of the singularity-free

controllers where the full motion is successfully
achieved.

## 5 Discussion

A comparison between the two main proposed meth-
ods in this article, quaternion- and dual quaternion-
based formation control, can also be made from a
qualitative perspective, focusing on the characteris-
tics of such definitions. From a task specification
and operator interface point of view, the quaternion-
based approach is more intuitive, as shape variables
such as formation size are defined as simple scalars.
Although somewhat more obscure, the representation
given by the dual quaternion method is more compact
and elegant from a mathematical point of view. From a
computational complexity perspective, the number of
variables to track is similar –16 variables for the two
4-DOF system in the examples– and both require the

**Fig. 20** Time lapse of the
controller implementing the
robot space definition in the
ROS/Gazebo Simulator
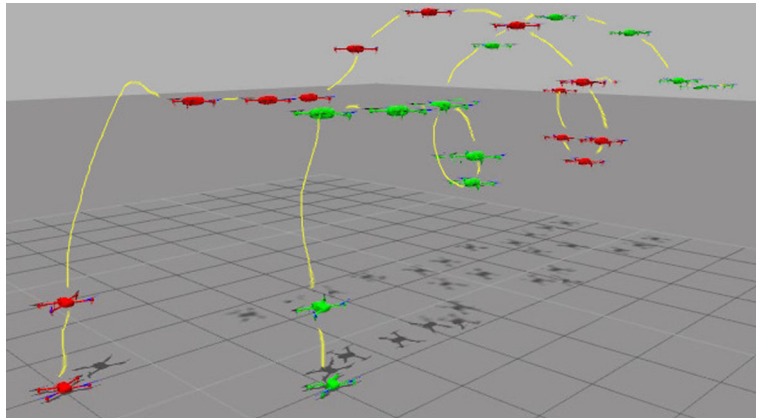
inverse of the Cluster Jacobian matrix to implement the control loop. In terms of tracking accuracy, there are no significant differences in the control architecture itself, and control performance will depend on the controller implemented in any particular case.

This article is focused on the implementation of formation control methods for the case of a group of two robots. Nevertheless, these approaches can be extended to formations of more robots. In particular, when extending the quaternion-based approach to a larger group, $x_c \in \mathbf{R}^3$ can still be the centroid of the robots' positions. An additional quaternion $\mathbf{q}_{ci}$ should be added for each additional robot in the formation, and the number of scalar shape parameters (as in the case of the proposed formation size $d_c$) should be increased to fully specify the formation. Similarly, in the case of the dual-quaternion approach, a dual quaternion $\hat{\mathbf{q}}_c$ would represent the position and orientation of the formation and additional dual quaternions would be included to represent the formation shape. It should be noted that the inclusion of each additional dual quaternion would result in a system where another 8 parameters must be tracked, as each dual quaternion is a 8-parameter expression. This may add to the computational complexity of the resulting system and therefore its scalability to a system with a large number of robots should be properly addressed.

## 6 Conclusions

Three solutions were presented to address the singularity problem when a minimal state representation is used in the cluster space control framework. First, a robot space controller that effectively creates robot space trajectories to be tracked by the controller provides a simple implementation for singularity-free control but lacks the 'well-behaved' motions naturally expected for the formation as it is unable to preserve the formation shape when following step inputs. Then, a singularity-free unit quaternion definition is proposed, adding redundancy to the system while keeping the abstraction concept of representing the formation through position, orientation and shape variables. Finally, the singularity issue is resolved proposing a dual quaternion definition that uses a redundant and compact description of the system at the expense of a more obscure representation. This representation introduces the novel concept of using

dual quaternions to represent formation attributes such as shape, together with position and orientation. Simulation results using MATLAB as well as models of the IRIS multicopters in the ROS/Gazebo open-source environment indicate the feasibility of the proposed solutions and show their differences and limitations. Future work will include extensions of the proposed methods to formations with more robots where additional shape parameters will be necessary to describe more complex group structures, as well as experimental verification employing a multi-UAV testbed.

## References

1. Yu, C.-H., Nagpal, R.: Biologically-inspired control for multi-agent self-adaptive tasks. In: AAAI (2010)
2. Lin, Z., Ding, W., Yan, G., Yu, C., Giua, A.: Leader–follower formation via complex laplacian. Automatica (2013)
3. Tan, K.-H., Lewis, M.A.: Virtual structures for high-precision cooperative mobile robotic control. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, vol. 1, pp. 132–139 (Nov 1996)
4. Ani Hsieh, M., Kumar, V., Chaimowicz, L.: Decentralized controllers for shape generation with robotic swarms. Robotica **26**(05), 691–701 (2008)
5. Mas, I., Kitts, C.: Dynamic control of mobile multirobot systems The cluster space formulation. Access, IEEE **2**, 558–570 (2014)
6. Mas, I., Li, S., Acain, J., Kitts, C.: Entrapment/escorting and patrolling missions in multi-robot cluster space control, pp. 5855–5861 (2009)
7. Mas, I., Kitts, C.: Object manipulation using cooperative mobile multi-robot systems. Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2012, WCECS 2012, 324–329 (2012)
8. Kitts, C.A., Mas, I.: Cluster space specification and control of mobile multirobot systems. IEEE/ASME Trans. Mechatron. **14**(2), 207–218 (April 2009)
9. Huizenga, F.: Singularity avoidance for the cluster space control of mobile multi-robot systems. Santa Clara University, Master's thesis (May 2012)
10. Belta, C., Kumar, V.: Motion generation for formations of robots: a geometric approach. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, 2001, vol. 2, pp. 1245–1250. IEEE (2001)
11. Fierro, R., Das, A., Spletzer, J., Esposito, J., Kumar, V., Ostrowski, J., Pappas, G., Taylor, C., Hur, Y., Alur, R., Lee, I., Grudic, G., Southall, B.: A framework and architecture for multi-robot coordination. The International Journal of Robotics Research **21**(10-11), 977–995 (2002)
12. Chaimowicz, L., Sugar, T., Kumar, V.: Mario Fernando Montenegro Campos. An architecture for tightly coupled multi-robot cooperation. In: Proceedings 2001 ICRA, IEEE International Conference on Robotics and Automation, 2001, vol. 3, pp. 2992–2997. IEEE (2001)

13. Maithripala, D.H.S., Berg, J.M., Maithripala, D.H.A., Jayasuriya, S.: A geometric virtual structure approach to decentralized formation control. In: American Control Conference (ACC) 2014, pp. 5736–5741. IEEE (2014)

14. Chaimowicz, L., Kumar, V.: Aerial shepherds: Coordination among uavs and swarms of robots. In: Distributed Autonomous Robotic Systems 6, pp. 243–252. Springer (2007)

15. Jia, Q., Li, G., Lu, J.: Formation control and attitude cooperative control of multiple rigid body systems. In: Sixth International Conference on Intelligent Systems Design and Applications 2006 ISDA'06, vol. 2, pp. 82–86. IEEE (2006)

16. Wang, P.K.C., Hadaegh, F.Y., Lau, K.: Synchronized formation rotation and attitude control of multiple free-flying spacecraft. J. Guid. Control. Dyn. **22**(1), 28–35 (1999)

17. Wang, X., Han, D., Changbin, Y., Zheng, Z.: The geometric structure of unit dual quaternion with application in kinematic control. J. Math. Anal. Appl. **389**(2), 1352–1364 (2012)

18. Dooley, J.R., McCarthy, J.: On the geometric analysis of optimum trajectories for cooperating robots using dual quaternion coordinates. In: Proceedings., 1993 IEEE International Conference on Robotics and Automation, 1993, vol. 1, pp. 1031–1036 (1993)

19. Han, D.-P., Wei, Q., Li, Z.-X.: Kinematic control of free rigid bodies using dual quaternions. Int. J. Autom. Comput. **5**(3), 319–324 (2008)

20. Wang, X., Yu, C., Lin, Z.: A dual quaternion solution to attitude and position control for rigid-body coordination. IEEE Trans. Robot. **28**(5), 1162–1170 (2012)

21. Dong, H., Qinglei, H., Ma, G.: Dual-quaternion based fault-tolerant control for spacecraft formation flying with finite-time convergence ISA transactions (2016)

22. Jinjie, W., Liu, K., Gao, Y., Zhang, B.: 6dof quasi-optimal integral sliding mode control for satellite formation flying using dual quaternion. In: Control and Decision Conference (CCDC), 2013 25th Chinese, pp. 1764–1769 (2013)

23. Yinqiu, W., Yu, F., Yu, C.: Distributed attitude and position consensus for networked rigid bodies based on unit dual quaternion. In: Control Conference (CCC), 2015 34th Chinese, pp. 7368–7373. IEEE (2015)

24. Kristiansen, R., Nicklasson, P.J., Gravdahl, J.T.: Quaternion-based backstepping control of relative attitude in a spacecraft formation. In: 2006 45th IEEE Conference on Decision and Control, pp. 5724–5729. IEEE (2006)

25. Adorno, B.V., Fraisse, P., Druon, S.: Dual position control strategies using the cooperative dual task-space framework. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), p. 2010 (2010)

26. Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., Von Stryk, O.: Comprehensive simulation of quadrotor uavs using ros and gazebo. In: Simulation, Modeling, and Programming for Autonomous Robots, pp. 400–411. Springer (2012)

27. Furrer, F., Burri, M., Achtelik, M., Roland Siegwart.: Robot Operating System (ROS): The Complete Reference (Volume 1), chapter RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. Springer International Publishing, Cham (2016)

**Ignacio Mas** received the Engineering degree in Electrical Engineering from the Universidad de Buenos Aires, Buenos Aires, Argentina, and the Ph.D. degree in Mechanical Engineering from Santa Clara University, Santa Clara, CA, USA. He was a Satellite Systems Engineer and Spacecraft Systems Designer at NASA Ames Research Center. Dr. Mas is currently an Assistant Researcher at the Argentine National Scientific and Technical Research Council (CONICET) and an Assistant Professor at the Instituto Tecnologico de Buenos Aires (ITBA), Buenos Aires, Argentina. His research interests include multirobot systems, coordinated navigation, and robot formation control.

**Christopher Kitts** is an Associate Professor of Mechanical Engineering, the Director of the Robotic Systems Laboratory, and an Associate Dean in the School of Engineering at Santa Clara University. He received the B.S.E. degree from Princeton University, Princeton, NJ, USA, the M.P.A. degree from the University of Colorado, Colorado Springs, CO, USA, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, USA. He has a decade of industry experience as a NASA contractor, as an officer and satellite constellation mission controller in the U.S. Air Force, as a Department of Defense Research Fellow, and as the Graduate Student Director of the Space Systems Development Laboratory, Stanford University. Dr. Kitts is a Fellow of the American Society of Mechanical Engineers.