2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Orientation in Cartesian Space Dynamic Movement Primitives

Aleš Ude[1,2], Bojan Nemec[1], Tadej Petrič[1,2], and Jun Morimoto[2]

***Abstract*— Dynamic movement primitives (DMPs) were proposed as an efficient way for learning and control of complex robot behaviors. They can be used to represent point-to-point and periodic movements and can be applied in Cartesian or in joint space. One problem that arises when DMPs are used to define control policies in Cartesian space is that there exists no minimal, singularity-free representation of orientation. In this paper we show how dynamic movement primitives can be defined for non minimal, singularity free representations of orientation, such as rotation matrices and quaternions. All of the advantages of DMPs, including ease of learning, the ability to include coupling terms, and scale and temporal invariance, can be adopted in our formulation. We have also proposed a new phase stopping mechanism to ensure full movement reproduction in case of perturbations.**

I. Introduction

With a focus on specifying goal directed robot behaviors, Ijspeert et al. [1], [2] proposed to model robot movements with dynamic movement primitives (DMPs). In their basic form, DMPs consist of a linear second order attractor system (differential equation) with the added nonlinear forcing term, which is applied to adapt the simple second order attractor dynamics to the specific robot movement. DMPs can be used to represent both periodic and discrete (point-to-point) movements. They have many favorable properties, e. g. they contain open parameters that can be used for learning without affecting the overall convergence and stability of the system, they can control timing without requiring an explicit time representation, they are robust against perturbations and can be modulated to adapt to different requirements.

While methods used for trajectory planing in joint space can often also be used in Cartesian space, the planing of orientation trajectories is sometimes more complicated. Unlike position, which can be represented by a 3-D vector, the robot's orientation cannot be represented this way because the set of all orientations, which we denote by $\mathrm{SO}(3)$, is not a vector space. $\mathrm{SO}(3)$, however, is a group and a real three dimensional manifold [3]. While $\mathrm{SO}(3)$ can be parameterized by three parameters, e. g. Euler angles, such parameterizations always contain singularities. It turns out that there exists no minimal (3-D) representation of orientation that 1. contains no singularities, i. e., a continuous orientational motion in 3-D space is continuous also in the 3-D parameter space, and 2. is continuously differentiable, i. e. the partial derivatives of the parameters with respect to any differential rotation, at any orientation, are finite.

[1]Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. ales.ude@ijs.si

[2]Dept. of Brain Robot Interface, ATR Computational Neuroscience Laboratories, Kyoto, Japan.

It is preferable to use parameterizations that do not introduce artificial discontinuities not existing in the real world into trajectory planing. As outlined above, such representations of orientation are non minimal and must therefore fulfill additional constraints in a higher dimensional space to be guaranteed to lie on the constraint manifold. This causes problems when integrating differential equations on $\mathrm{SO}(3)$ because general integration methods do not have any information about the structure of $\mathrm{SO}(3)$ and can cause the integrated parameters to depart from the constraint manifold.

In this paper we first propose a new approach that avoids integrating the orientation parameters directly and is therefore guaranteed to generate parameters that lie in $\mathrm{SO}(3)$. The proposed method can use either rotation matrices or quaternions. We compare it to the approach previously proposed in [4] and point out the difference between quaternion and rotation matrix based approach. We then show that different extensions of DMPs regarding modulation and coupling can be adopted for orientation DMPs and propose a new approach to phase stopping in case of external perturbations.

II. Movement Representation in Cartesian Space

The basic idea of dynamic movement primitives (DMPs) is to model movements by a system of differential equations with well understood stability properties. This system should ensure the desired behavior, e. g. convergence to the specified attractor point [1], [2]. A nonlinear forcing term, which allows the modeling of more complex movements, is added to the basic system of differential equations. A DMP for a single degree of freedom point-to-point movement y is defined by the following set of nonlinear differential equations [1]

$$\tau \dot{z} = \alpha_z(\beta_z(g-y)-z)+f(x), \quad (1)$$

$$\tau \dot{y} = z, \quad (2)$$

where x is the phase variable, z is the scaled velocity of the movement, g is the desired final position on the movement, and f is a nonlinear forcing term. With the choice $\tau > 0$ and $\alpha_z = 4\beta_z$, the linear part of equation system (1) – (2) becomes critically damped and $y,\ z$ monotonically converge to a unique attractor point at $y = g$, $z = 0$ [1]. $f(x)$ is defined as a linear combination of N nonlinear radial basis functions, which enables the robot to follow any smooth trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x(g-y_0), \quad (3)$$

$$\Psi_i(x) = \exp\left(-h_i\,(x-c_i)^2\right). \quad (4)$$

c_i are the centers of Gaussians distributed along the phase of the movement and h_i their widths. The phase x has been introduced to avoid explicit time dependency. Its evolution follows the first order linear dynamics

$$\tau \dot{x} = -\alpha_x x, \tag{5}$$

where $\alpha_x > 0$ and x(0) = 1. For a given N and setting the time constant τ equal to the total duration of the desired movement, we can define $c_i = \exp\left(-\alpha_x \frac{i-1}{N-1}\right)$, $h_i = \frac{1}{(c_{i+1} - c_i)^2}$, $h_N = h_{N-1}, i = 1, \ldots, N$. For each Cartesian degree of freedom, the weights w_i should be adjusted so that the desired behavior is achieved. To control a robot with more than one degree of freedom, we represent the trajectory of every degree of freedom with its own equation system (1) – (2), but with the common phase (5) to synchronize them.

A. Rotation Matrix Based DMPs

The basics for the developments in this section are provided in Appendix I. To fully describe a movement of the robot's end effector in Cartesian space, we can specify its position trajectory $\mathbf{p}(t) \in \mathbb{R}^3$ and orientation trajectory $\mathbf{R}(t) \in \mathbb{R}^{3\times3}$. Each of the 12 parameters could be represented by its own DMP system (1) – (2). However, the coefficients of the orientation trajectory $\mathbf{R}(t)$ are not independent of each other, thus if we integrate them independently, the integrated rotation matrix will gradually start deviating from the true rotation matrix. We therefore reformulate Eq. (1) – (2) as follows

$$\tau \dot{\boldsymbol{\eta}} = \alpha_z(\beta_z \log(\mathbf{R}_g \mathbf{R}^{\mathrm{T}}) - \boldsymbol{\eta}) + \mathbf{f}_o(x). \tag{6}$$

$$\tau \dot{\mathbf{R}} = [\boldsymbol{\eta}]_\times \mathbf{R}, \tag{7}$$

where $\mathbf{R}_g$ denotes the goal orientation. The motivation to write (2) as (7) is provided by Eq. (40) from Appendix I. Thus $\boldsymbol{\eta}$ is just the scaled angular velocity $\boldsymbol{\omega}$. The motivation for (6) is the fact that, as evident from Eq. (44), $\boldsymbol{\omega} = \log(\mathbf{R}_g \mathbf{R}^{\mathrm{T}})$ corresponds to the angular velocity that rotates matrix $\mathbf{R}$ into the goal orientation $\mathbf{R}_g$ within the unit time. This is equivalent to the difference $g - y$ in (1), which computes the linear velocity that translates parameter y to g within the unit time. See Appendix I for more details about rotation matrix logarithm and its relation to the angular velocity. The nonlinear forcing term

$$\mathbf{f}_o(x) = \mathbf{D}_o \frac{\sum_{i=1}^N \mathbf{w}_i^o \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x \tag{8}$$

contains free parameters $\mathbf{w}_i^o \in \mathbb{R}^3$, which need to be determined to follow any given rotation trajectory. The data for computing them is usually gathered by sampling along the trajectory, where the trajectory is obtained either by human demonstration or by guiding the robot arm through a desired trajectory and recording the resulting orientations $\mathbf{R}_j$, velocities $\boldsymbol{\omega}_j$, and accelerations $\dot{\boldsymbol{\omega}}_j$ at sampling times t_j. The parameters $\mathbf{w}_i^o$ are calculated by solving the following system of linear equations, which we obtain from (6)

$$\frac{\sum_{i=1}^N \mathbf{w}_i^o \Psi_i(x_j)}{\sum_{i=1}^N \Psi_i(x_j)} x_j = \mathbf{D}_o^{-1}\left(\tau \dot{\boldsymbol{\eta}}_j + \alpha_z \boldsymbol{\eta}_j - \alpha_z \beta_z \left(\log\left(\mathbf{R}_g \mathbf{R}_j^{\mathrm{T}}\right)\right)\right), \tag{9}$$

where phases x_j are calculated by integrating (5), i. e.

$$x_j = x(t_j) = \exp\left(-\frac{\alpha_x}{\tau} t_j\right) \tag{10}$$

and $j = 0, \ldots, T$. The relation between the rotation matrix derivative and angular velocity is given by (40), thus from (7) we obtain $[\boldsymbol{\eta}]_\times = \tau[\boldsymbol{\omega}]_\times$, which means that $\boldsymbol{\eta}_j = \tau \boldsymbol{\omega}_j$, $\dot{\boldsymbol{\eta}}_j = \tau \dot{\boldsymbol{\omega}}_j$. The scaling factor $\mathbf{D}_o = \mathrm{diag}\left(\log\left(\mathbf{R}_g \mathbf{R}_0^{\mathrm{T}}\right)\right) \in \mathbb{R}^{3\times3}$ is a diagonal matrix built from a 3-D vector, where the coefficients of the vector are used to define the diagonal elements of the matrix. The modulation by $\mathbf{D}_o$ leads to useful scaling properties of the DMP system when the goal configuration $\mathbf{g}_o$ and consequently the amplitude of the movement change (see also Section V).

Standard Euler formulas can be applied to integrate (6). To integrate (7) we use the expression obtained from (44)

$$\mathbf{R}(t + \Delta t) = \exp\left(\Delta t \frac{[\boldsymbol{\eta}]_\times}{\tau}\right) \mathbf{R}(t), \tag{11}$$

which is guaranteed to generate a rotation matrix because the exponential map $\exp(\Delta t [\boldsymbol{\eta}]_\times / \tau)$ results in a rotation matrix and the product of two rotation matrices is a rotation matrix.

B. Position Trajectories

For the sake of completeness we also rewrite DMP equations (1) – (2), which are used to encode the positional part of the trajectory, in vector form

$$\tau \dot{\mathbf{z}} = \alpha_z(\beta_z(\mathbf{g}_p - \mathbf{p}) - \mathbf{z}) + \mathbf{f}_p(x), \tag{12}$$

$$\tau \dot{\mathbf{p}} = \mathbf{z}, \tag{13}$$

where $\mathbf{g}_p \in \mathbb{R}^3$ denotes the final position on the recorded trajectory. The forcing term $\mathbf{f}_p$ is defined as

$$\mathbf{f}_p(x) = \mathbf{D}_p \frac{\sum_{i=1}^N \mathbf{w}_i^p \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x, \tag{14}$$

where $\mathbf{D}_p = \mathrm{diag}\left(\mathbf{g}_p - \mathbf{p}_0\right) \in \mathbb{R}^{3\times3}$. As mentioned above, the diagonal matrices $\mathbf{D}_p$ in (14) and $\mathbf{D}_o$ in (8) are used to scale the movement amplitude if the goal configuration $\mathbf{g}_p$ and/or $\mathbf{g}_o$ change. To track the desired Cartesian space trajectories, we need to integrate (12) – (13) and (6) – (7) along with the common phase (5).

Given the robot tool center point trajectory $\{\mathbf{p}_j, \dot{\mathbf{p}}_j, \ddot{\mathbf{p}}_j, t_j\}_{j=0}^T$, the free parameters can be calculated in a similar way as in (9), i. e. by first rewriting (12) – (13) as one second order differential equation [1] and then solving the resulting system of linear equations

$$\frac{\sum_{i=1}^N \mathbf{w}_i^p \Psi_i(x_j)}{\sum_{i=1}^N \Psi_i(x_j)} x_j = \mathbf{D}_p^{-1}\left(\tau^2 \ddot{\mathbf{p}}_j + \alpha_z \tau \dot{\mathbf{p}}_j - \alpha_z \beta_z \left(\mathbf{g}_p - \mathbf{p}\right)\right), \tag{15}$$

where the phases x_j are calculated as in (10).

III. Quaternion Based DMPs

An alternative to rotation matrices is provided by unit quarternions $\mathbf{q} = v + \mathbf{u} \in \mathrm{S}^3$, where S^3 is a unit sphere in $\mathbb{R}^4,\ v \in \mathbb{R},\ \mathbf{u} \in \mathbb{R}^3$. Quaternions also provide a singularity-free, non minimal representation of the orientation, but with only four parameters compared to nine of rotation matrices. They have been utilized to represent orientation in various contexts including robot control [5], trajectory modeling [6], and tracking [7], [8]. This representation is not unique because unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same orientation. We can transform the rotational DMP equations (6) – (7) into the quaternion form as follows

$$\begin{aligned} \tau\dot{\boldsymbol{\eta}} &= \alpha_z\left(\beta_z 2\log\left(\mathbf{g}_o * \overline{\mathbf{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(x), && (16)\\ \tau\dot{\mathbf{q}} &= \frac{1}{2}\boldsymbol{\eta} * \mathbf{q}, && (17) \end{aligned}$$

where $\mathbf{g}_o \in \mathrm{S}^3$ denotes the goal quaternion orientation, bar denotes the quaternion conjugation defined as $\overline{\mathbf{q}} = \overline{v + \mathbf{u}} = v - \mathbf{u}$, and $*$ denotes the quaternion product

$$\begin{aligned} \mathbf{q}_1 * \mathbf{q}_2 &= (v_1 + \mathbf{u}_1) * (v_2 + \mathbf{u}_2) && (18)\\ &= (v_1 v_2 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_2) + (v_1\mathbf{u}_2 + v_2\mathbf{u}_1 + \mathbf{u}_1 \times \mathbf{u}_2). \end{aligned}$$

$\boldsymbol{\eta} \in \mathbb{R}^3$ is treated as quaternion with 0 scalar part in (17). The quaternion logarithm $\log : \mathrm{S}^3 \mapsto \mathbb{R}^3$ is defined as [7]

$$\log(\mathbf{q}) = \log(v + \mathbf{u}) = \begin{cases} \arccos(v)\dfrac{\mathbf{u}}{\|\mathbf{u}\|}, & \mathbf{u} \neq 0 \\ [0,0,0]^{\mathrm{T}}, & \text{otherwise} \end{cases}. \quad (19)$$

The basis for the above equations is provided by the facts that firstly, given two unit quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ specifying two different orientations, the rotation from $\mathbf{q}_1$ to $\mathbf{q}_2$ is specified by $\Delta\mathbf{q} = \mathbf{q}_2 * \overline{\mathbf{q}}_1$, and secondly, by formula (23). Unlike with rotation matrices (see Appendix I), the logarithmic map defined on S^3 has no discontinuity boundary, just a singularity at a single quaternion $\mathbf{q} = -1 + [0,0,0]^{\mathrm{T}}$. It can be used to specify a distance metric on S^3 [7]

$$\mathrm{d}(\mathbf{q}_1, \mathbf{q}_2) = \begin{cases} 2\pi, & \mathbf{q}_1 * \overline{\mathbf{q}}_2 = -1 + [0,0,0]^{\mathrm{T}} \\ 2\|\log(\mathbf{q}_1 * \overline{\mathbf{q}}_2)\|, & \text{otherwise} \end{cases}. \quad (20)$$

Note that d is not a metric on $\mathrm{SO}(3)$ because $\mathrm{d}(\mathbf{q}, -\mathbf{q}) = 2\pi$, although $\mathbf{q},\ -\mathbf{q}$ represent the same orientation. The nonlinear forcing term $\mathbf{f}_o$ is defined as in (8). It contains free parameters $\mathbf{w}_i^o \in \mathbb{R}^3$, which can be used to encode any sampled orientation trajectory $\{\mathbf{q}_j, \boldsymbol{\omega}_j, \dot{\boldsymbol{\omega}}_j, t_j\}_{j=0}^T$ with the quaternion based DMP defined by (16) – (17) and (5). Parameters $\mathbf{w}_i^o$ are obtained by solving the following system of linear equations

$$\begin{aligned} &\frac{\sum_{i=1}^N \mathbf{w}_i^o \Psi_i(x_j)}{\sum_{i=1}^N \Psi_i(x_j)} x_j = \\ &\mathbf{D}_o^{-1}\left(\tau\dot{\boldsymbol{\eta}}_j - \alpha_z\left(\beta_z 2\log\left(\mathbf{g}_o * \overline{\mathbf{q}}_j\right) - \boldsymbol{\eta}_j\right)\right), \end{aligned} \quad (21)$$

where phases x_j are obtained as in (10). The relation between the quaternion derivative and angular velocity is given by $\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega} * \mathbf{q}$, thus from (17) we obtain $\boldsymbol{\eta}_j = \tau\boldsymbol{\omega}_j$, $\dot{\boldsymbol{\eta}}_j = \tau\dot{\boldsymbol{\omega}}_j$.

Unlike in our work, Pastor et al. [4] used in Eq. (16) just the vector part of the quaternion product $v + \mathbf{u} = \mathbf{g}_o * \overline{\mathbf{q}}$, i. e. $\mathbf{u} \in \mathbb{R}^3$, instead of $2\log\left(\mathbf{g}_o * \overline{\mathbf{q}}\right)$. A similar type of error was used also in [9]. Defining $\mathbf{u} = \mathrm{vec}\left(\mathbf{g}_o * \overline{\mathbf{q}}\right)$, their system can be written as

$$\tau\dot{\boldsymbol{\eta}} = \alpha_z\left(\beta_z \mathrm{vec}\left(\mathbf{g}_o * \overline{\mathbf{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(x). \quad (22)$$

While this is equivalent as far as the direction of change is concerned, such a formulation does not fully take into account the geometry of $\mathrm{SO}(3)$. It holds namely

$$\boldsymbol{\omega} = 2\,\log\left(\mathbf{g}_o * \overline{\mathbf{q}}\right), \quad (23)$$

where $\boldsymbol{\omega}$ denotes the angular velocity that rotates quaternion $\mathbf{q}$ into $\mathbf{g}_o$ within unit time. Thus only the logarithmic map multiplied by 2 can provide proper mapping of the quaternion product $\mathbf{g}_o * \overline{\mathbf{q}}$ onto the angular velocity. Another difference is that there is no scaling factor $\mathbf{D}_o$ in [4]. See Section V for the analysis of differences between the two approaches.

For the integration of (17) we can use the formula

$$\mathbf{q}(t + \Delta t) = \exp\left(\frac{\Delta t}{2}\frac{\boldsymbol{\eta}(t)}{\tau}\right) * \mathbf{q}(t). \quad (24)$$

This is the quaternion version of (11), where the quaternion exponential is defined as [7]

$$\exp(\mathbf{r}) = \begin{cases} \cos\left(\|\mathbf{r}\|\right) + \sin\left(\|\mathbf{r}\|\right)\dfrac{\mathbf{r}}{\|\mathbf{r}\|}, & \mathbf{r} \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (25)$$

If we limit the domain of the exponential map $\exp : \mathbb{R}^3 \mapsto \mathrm{S}^3$ to $\|\mathbf{r}\| < \pi$ and the domain of the logarithmic map to $\mathrm{S}^3/(-1 + [0,0,0]^{\mathrm{T}})$, then both mappings become one-to-one, continuously differentiable and inverse to each other.

IV. Modulations of Orientation DMPs

One of the major advantages of DMPs is that various useful behaviors can be achieved by manipulating the underlying differential equation systems. In this section we address the issue of phase stopping, goal switching, and robustness against perturbations in the context of quaternion based DMPs. The derivations for rotation matrices are equivalent. The extensions in this section are applied only during the execution of the movement. In the training phase, the original DMP equations should be used.

A. Phase Stopping

The reason for expressing the phase with a differential equation is that its time evolution can be manipulated through this differential equation. In contrast, an explicit time representation cannot be manipulated as easily [2]. For DMP phase stopping [10], the original equation for phase (5) can be replaced with

$$\tau\dot{x} = -\frac{\alpha_x x}{1 + \alpha_{px}\left(\|\tilde{\mathbf{p}} - \mathbf{p}\| + \gamma\,\mathrm{d}(\tilde{\mathbf{q}}, \mathbf{q})\right)}, \quad (26)$$

where $\|\tilde{\mathbf{p}} - \mathbf{p}\| + \gamma\,\mathrm{d}(\tilde{\mathbf{q}}, \mathbf{q})$ is the trajectory tracking error, $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ are the actual position and orientation of the tool center point, respectively, and $\mathbf{p}$ and $\mathbf{q}$ the corresponding DMP

control outputs. Note that in the case of large tracking errors, the error value $\|\tilde{\mathbf{p}} - \mathbf{p}\| + \gamma\, \mathrm{d}(\tilde{\mathbf{q}}, \mathbf{q})$ becomes large which in turn makes the phase change $\dot{x}$ small. Thus the phase evolution is stopped until the robot reduces the tracking error. Ijspeert et al. [10] proposed to modify also Eq. (2) to ensure faster error reduction

$$\tau \dot{y} = z + \alpha_{py}(\tilde{y} - y), \tag{27}$$

where $\tilde{y}$ denotes the actual position of the robot and y the DMP calculated position. In the context of Cartesian space DMPs, Eq. (27) becomes different for the positional and orientational part of the trajectory, which are respectively encoded by Eq. (13) and (7). We obtain

$$\tau \dot{\mathbf{p}} = \mathbf{z} + \alpha_{pp}(\tilde{\mathbf{p}} - \mathbf{p}), \tag{28}$$

$$\tau \dot{\mathbf{q}} = \frac{1}{2}\left(\boldsymbol{\eta} + \alpha_{pr} 2 \log\left(\tilde{\mathbf{q}} * \overline{\mathbf{q}}\right)\right) * \mathbf{q}, \tag{29}$$

Eq. (29) is integrated using a formula analog to (24).

B. Goal Switching

When the goal orientation $\mathbf{g}_o$ is switched to a new orientation, the discontinuity that arises in $\log(\mathbf{g}_o * \overline{\mathbf{q}})$ used in (16) causes a discontinuity in the angular acceleration $\dot{\boldsymbol{\omega}}$. For the standard DMPs, Ijspeert et al. [1] suggested to avoid this discontinuity by adding an additional equation to the differential equation system (1) – (2), which causes the goal g in (1) to smoothly transition to the new goal g_0

$$\tau \dot{g} = \alpha_g(g_0 - g). \tag{30}$$

Note that what used to be a constant g is now a continuous variable, while g_0 can change discontinuously, based for example on the external sensory information. In the context of quaternion based DMPs, this equation becomes

$$\tau \dot{\mathbf{g}}_o = \alpha_{go} \log(\mathbf{g}_{o,new} * \overline{\mathbf{g}}_o) * \mathbf{g}_o. \tag{31}$$

Equivalently to g in (30), $\mathbf{g}_o$ is now a variable that continuously transitions to a new goal $\mathbf{g}_{o,new}$. Equation (31) should be integrated together with (16) – (17) using formula (24).

C. Robustness to Perturbations through Coupling of DMPs

Another advantage of DMPs is that they are robust against perturbations [1], e. g. when a robot is physically pushed away from the desired trajectory while moving. The standard approach to perturbations is to simply continue integrating Eq. (5), (16) – (17) after the perturbation has arisen. This way the robot gradually returns to the desired trajectory. However, this approach does not ensure that the complete movement is reproduced, just that the robot returns to the original movement. We propose to apply the phase stopping mechanism when it is necessary to execute the complete movement. For this purpose we enhance Eq. (26), (28), (29) as follows

$$\tau \dot{x} = -\frac{\alpha_x x}{1 + \alpha_{px}\varepsilon(\tilde{\mathbf{p}}, \mathbf{p}_d, \mathbf{p}, \tilde{\mathbf{q}}, \mathbf{q}_d, \mathbf{q})}, \tag{32}$$

$$\tau \dot{\mathbf{p}} = \mathbf{z} + \alpha_{pp}(\tilde{\mathbf{p}} + \mathbf{p}_d - 2\mathbf{p}), \tag{33}$$

$$\tau \dot{\mathbf{q}} = \frac{1}{2}\left(\boldsymbol{\eta} + \alpha_{pr} 2 \left(\log\left(\tilde{\mathbf{q}} * \overline{\mathbf{q}}\right) + \log\left(\mathbf{q}_d * \overline{\mathbf{q}}\right)\right)\right) * \mathbf{q}, \tag{34}$$

where

$$\varepsilon = \|\tilde{\mathbf{p}} - \mathbf{p}\| + \|\mathbf{p}_d - \mathbf{p}\| + \gamma\, \left(\mathrm{d}(\tilde{\mathbf{q}}, \mathbf{q}) + \mathrm{d}(\mathbf{q}_d, \mathbf{q})\right),$$

$\mathbf{p}_d$ and $\mathbf{q}_d$ are the desired position and orientation, respectively, and all other variables and constants are as in (26), (28), (29). The unit quaternion distance metric d is defined in (20). The complete approach to control the robot is specified by the following coupled DMP system

- A DMP system obtained by adding (16), (12) to (33) – (34). Its outputs are used to generate motor commands.
- A separate DMP that uses standard DMP equations (16) – (17) and (12) – (13) with the same constants and variables as the above system. This DMP generates the desired position and orientation for (33) – (34).
- The coupling of both DMP systems occurs through the phase equation (32), which is used by both.

The proposed coupled system halts the phase evolution whenever the robot controller is unable to track the commanded position and orientation or whenever the commanded position and orientation deviate from the desired values. The second term in (33), (34) ensures that the system can recover once the perturbation is removed. After recovery the robot continues to perform the movement at the spot on the trajectory where it was before the onset of perturbation. Note that the original DMP system is equivalent to the one proposed in this section if the robot tracks the desired trajectory perfectly and there are no unexpected perturbations.

V. Experimental Results

We start by comparing the performance of the quaternion based DMPs to the method of Pastor et al. [4], which uses the quaternion difference vector $\mathrm{vec}\left(\mathbf{g}_o * \overline{\mathbf{q}}\right)$ as in (22), instead of $2\log\left(\mathbf{g}_o * \overline{\mathbf{q}}\right)$ as in (16). Fig. 1 and 2 show that our newly developed approach converges to the attractor point much faster than the approach proposed in [4], which is the desired characteristics of the linear part of the DMP system that should ensure convergence to the attractor point. Part of the reason for this is the lack of multiplication by 2 in (22). If we modify (22) to

$$\tau \dot{\boldsymbol{\eta}} = \alpha_z\left(\beta_z 2\mathrm{vec}\left(\mathbf{g}_o * \overline{\mathbf{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(x), \tag{35}$$

the DMP system (16) – (17) still generates a significantly quicker response than (35), (17), and converges faster, but the difference is smaller, as we can see by comparing Fig. 1 and 3. Nevertheless, the response of (16) – (17) is clearly preferable. Fig. 1 and 4 show that as expected, the quaternion based DMP (16) – (17) and the rotation matrix based DMP (6) – (7) generate exactly the same response in terms of angular velocity. In this experiment, we set $\mathbf{q}_0 = 0.3717 + [-0.4993, -0.6162, 0.4825]^{\mathrm{T}}$ as a starting orientation and $\mathbf{g}_o = 0.2471 + [0.1797, 0.3182, -0.8974]^{\mathrm{T}}$ as the goal orientation. The other constants were specified as follows: $\alpha_x = 2, \alpha_z = 48, \beta_z = 12, \tau = 3.5$.

We have thus shown theoretically and experimentally that the proposed systems (6) – (7) and (16) – (17) are significantly better than (35), (17), even though the multiplication

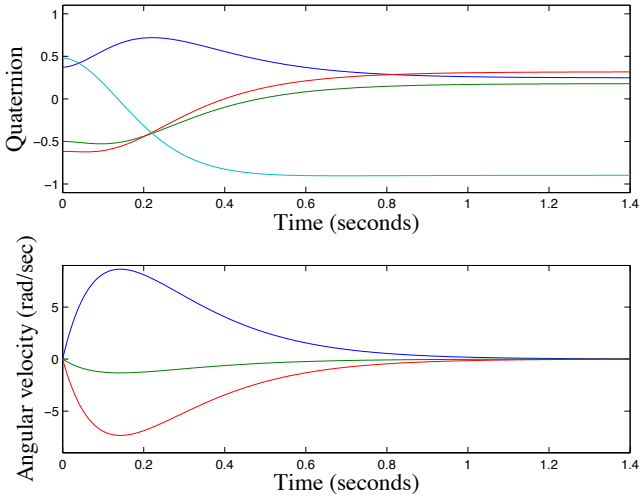

Fig. 1. Response of the DMP system (16) – (17) without nonlinear term $\mathbf{f}_o$ (this paper). The upper graph shows the time trajectories of the four unit quaternion components, and the lower graph the time trajectories of the three components of the angular velocity.

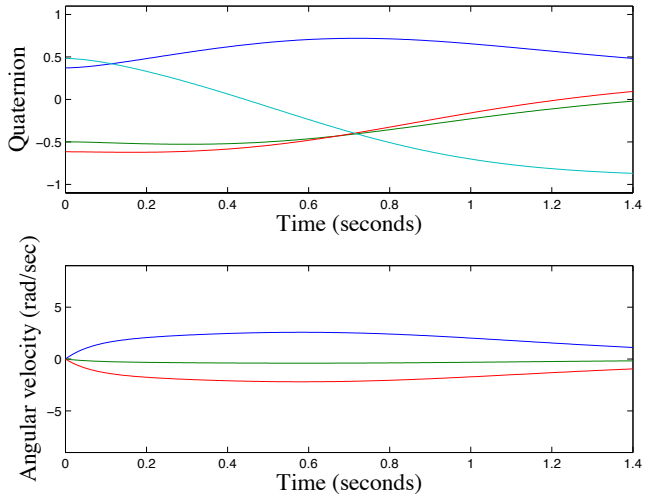

Fig. 2. Response of the DMP system (22), (17) with quaternion difference instead of the logarithmic map and without nonlinear term $\mathbf{f}_o$ (Pastor et al. [4]).

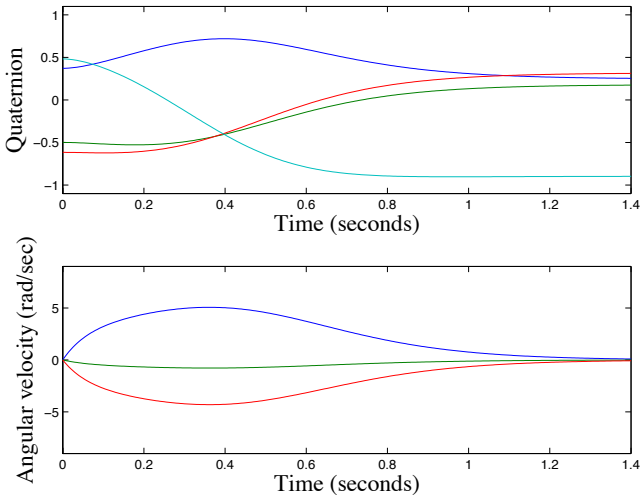

Fig. 3. Response of the DMP system (35), (17) with double quaternion difference instead of the logarithmic map and without nonlinear term $\mathbf{f}_o$.

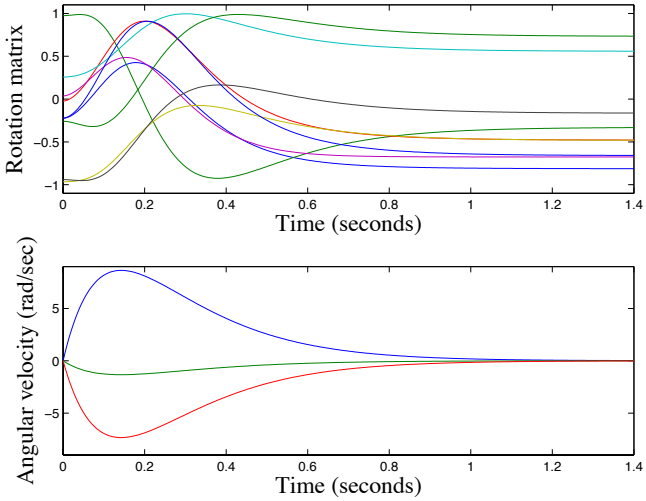

Fig. 4. Response of the rotation matrix based DMP system (6) – (7) without nonlinear term $\mathbf{f}_o$. The upper graph shows the time trajectories of the nine rotation matrix components, and the lower graph the time trajectories of the three components of the angular velocity.

by 2 has been added compared to the system (22), (17) from [4]. Note that 2 is the correct multiplier to obtain a critically damped system in (16) – (17). If we further increase the multiplier, the resulting system is not critically damped but starts oscillating towards the goal orientation, which is suboptimal for robot control.

In the next experiment we tested the approximation of the desired trajectories. We used exactly the same parameters as above except for τ, which was set to 1.5. To generate an example orientation trajectory we sampled a minimum jerk polynomial between start and end quaternion $\mathbf{q}_0$ and $\mathbf{g}_o$, respectively, and normalized the resulting quaternion trajectory. The results shown in Fig. 5 and 6 demonstrate that both approaches can accurately reproduce the desired trajectories. But this experiment also showed that there is a subtle issue that one should be aware of when using the rotation matrix based DMPs. Namely, the rotation matrix logarithm as defined in (43) has a discontinuity at $\|\log(\mathbf{R})\| = \pi$. This causes problems when estimating DMPs through equation system (9) because $\mathbf{f}_o$ can become discontinues when this boundary is crossed, even though the movement itself is not. As evident from Fig. 5, it is possible to resolve this problem by adding the appropriate constants to ensure the continuity of $\mathbf{f}_o$ in (6). On the other hand, the quaternion based DMPs do not suffer from this problem because there is no discontinuity boundary for quaternion logarithm (19) on the unit sphere S^3. The only singularity is at $\mathbf{q} = -1 + [0, 0, 0]^{\mathrm{T}}$, which is rarely or never hit in practice and can also easily be dealt with (for example, by assuming the previous direction vector in the logarithm when $-1 + [0, 0, 0]^{\mathrm{T}}$ is reached on the trajectory). For this reason we prefer to use quaternion representation to avoid the rather tedious bookkeeping issues.

Fig. 7 shows the performance of the phase stopping procedure described in Section IV-A. All parameters were the same as in the previous experiment except for τ, which was set to 2.5. As with standard DMPs, when the movement is stopped due to an external perturbation, the proposed

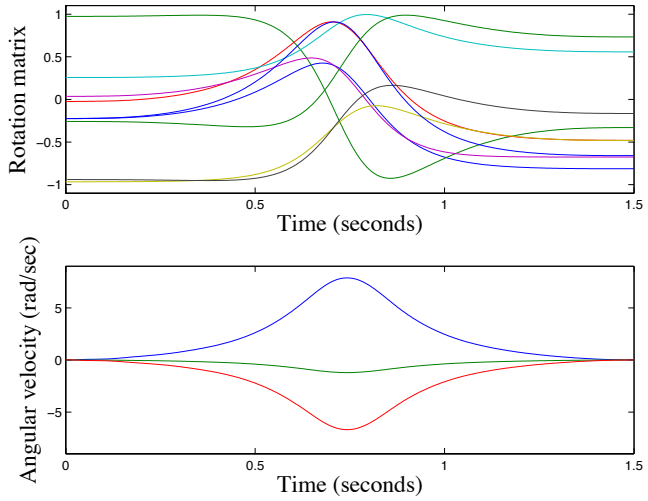

Fig. 5. Reproduction of the desired minimum jerk polynomial with the rotation matrix based DMP system (6) – (7).

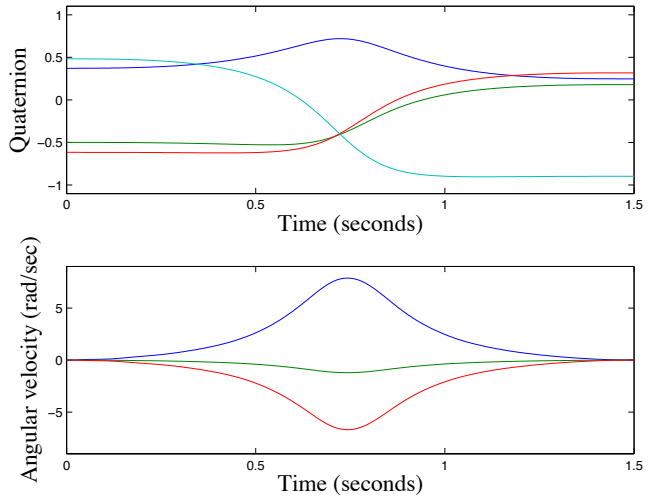

Fig. 6. Reproduction of the desired minimum jerk polynomial with the quaternion based DMP system (16) – (17)

approach halts the phase for the duration of the perturbation. Once the perturbation is removed, the trajectory quickly recovers – aided by the additional term in Eq. (29) – without any jumps. While such a behaviour could be realized with more standard techniques like splines, DMPs can achieve it without any bookkeeping regarding the timing and control of the desired movement. Simply integrating Eq. (26), (28), and (29) is enough to generate the appropriate response to the "movement stop" perturbation.

The response of the system to the goal change is shown in Fig. 8. Compared to the original movement of Fig. 6, the form of movement has been quite well preserved despite a fairly large change of the goal orientation. As expected, the change is larger in the angular velocity than in the quaternion trajectory. In all our experiments, the system (16), (17), (31) reliably converged to the new goal. The shape of the movement was also preserved most of the time, but in general the robustness of shape preservation was somewhat lower than in the case of position trajectories. The main reason for this is the more complicated geometry of $\mathrm{SO}(3)$ compared

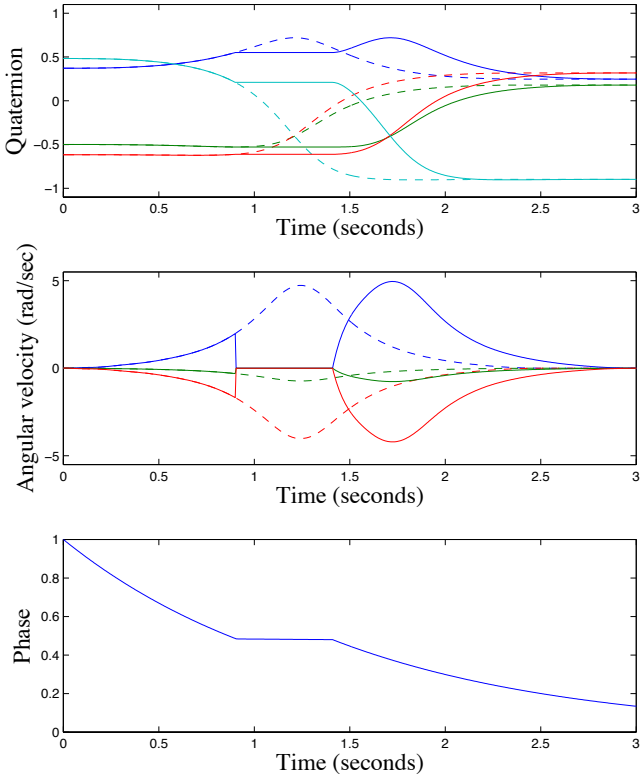

Fig. 7. The effects of phase stopping on quaternion based DMPs defined by (26), (28), (29). The original, unperturbed trajectory is indicated with the dashed lines, and the perturbed movement, which was stopped between 0.9 and 1.4 seconds, with full lines.

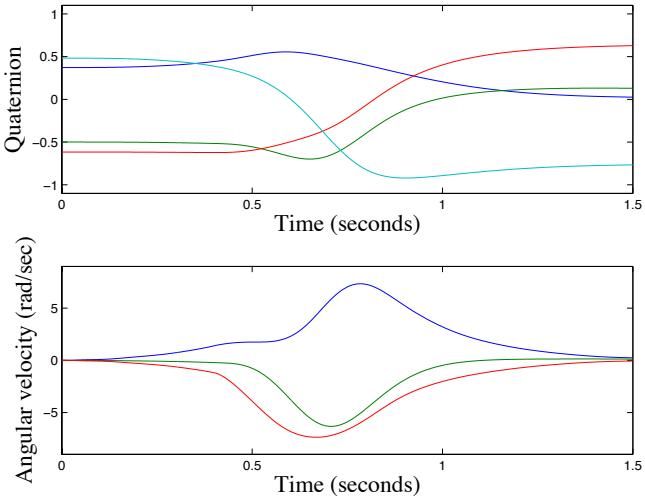

Fig. 8. Response of the DMP system (16), (17), (31) to the switching of goal orientation during the movement. The goal of the movement was changed at 0.4 seconds. The change of goal orientation corresponded to a rotation of 30.4 degrees. The original movement is shown in Fig. 6.

to the Euclidean space.

In the final simulation experiment we showed the robustness of the newly proposed phase stopping mechanism to sudden perturbations. As shown in Fig. 9, if we simply continue integrating the DMP equations (16), (17) after a perturbation, the movement converges back to the original movement and attains the desired goal orientation. However, the desired trajectory is not reproduced in its entirety. The

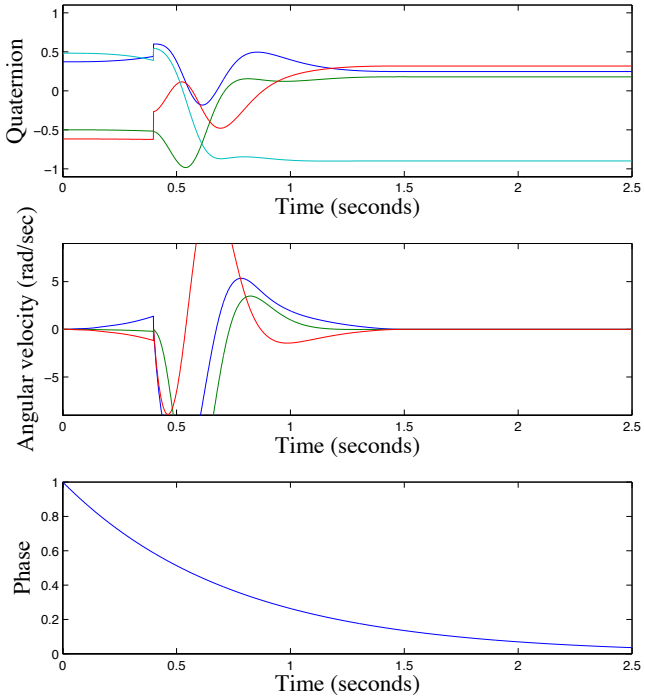

Fig. 9. Response of the DMP system (16), (17) to a perturbation. Here the orientation trajectory was perturbed by rotation of 28.3 degrees. The angular velocity was simultaneously set to 0. The graph shows the convergence of the perturbed movement to the final goal orientation.

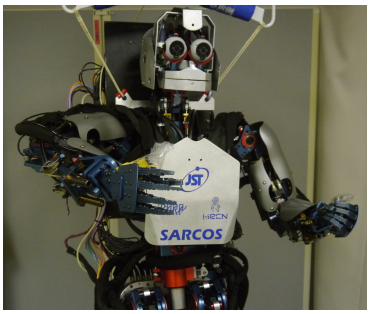

Fig. 10. Humanoid robot CBi [11] while following a Cartesian space DMP with its right arm.

complete movement reproduction can be achieved by the proposed coupled phase stopping DMP of Section IV-C. Fig. 11 shows that with this system the phase is stopped after the perturbation, which occurs due to large discrepancies between the desired and actual orientation on the trajectory. The robot thus has time to recover from the perturbation and return back to the desired trajectory, which is provided by the coupled DMP. Once the commanded orientation is close enough to the desired orientation, the robot resumes moving along the desired trajectory until it reaches the final configuration. Which of the two behaviors is more favorable depends on the requirements of the task.

We have also tested the proposed DMP system on a real robot. The generated trajectory is shown in Fig. 12.

VI. Conclusion

In this paper we proposed a new approach to specify orientation in Cartesian space dynamic movement primitives. The main contributions are

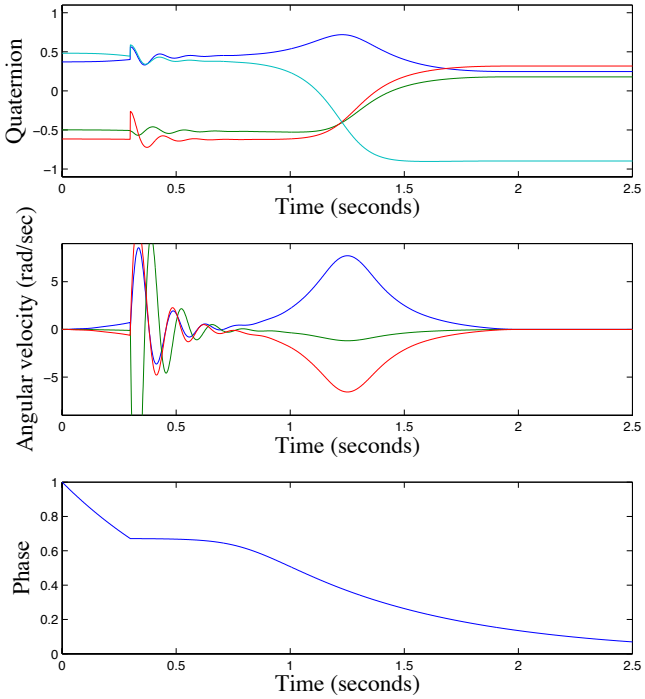

Fig. 11. Response of the phase stopping DMP (16), (34), (32) coupled with the standard DMP (16), (17) to the same perturbation as in Fig. 9. The graph shows the recovery of the movement back to the desired movement, which was the same as in Fig. 6.

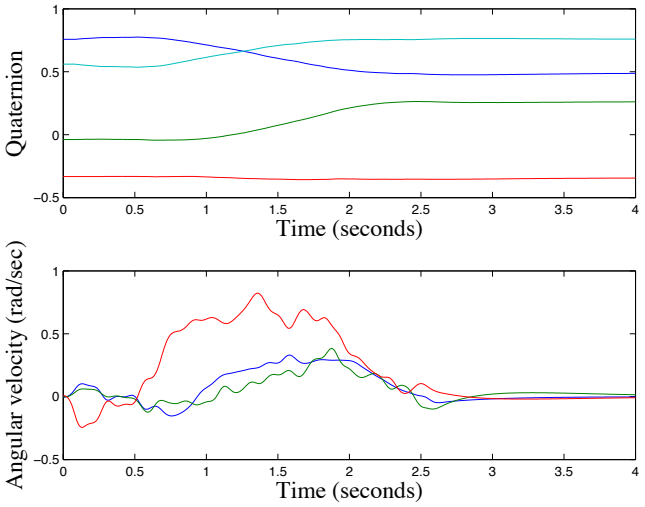

Fig. 12. Response of the DMP system on a real robot.

- The application of logarithmic map to represent orientation difference in Cartesian space DMPs based on quaternions or rotation matrices (see Section II and III). The newly proposed system has significantly better attractor properties than the previously proposed approach based on the quaternion difference vector.
- Theoretical and experimental analysis that showed that quaternion based DMPs are easier to use than rotation matrix based DMPs due to the discontinuity boundary that exists for the logarithmic map defined on rotation matrices, which does not exist for the logarithmic map defined on the space of unit quaternions.

- The formulation of many standard DMP extensions (phase stopping, goal switching, scaling) for the case of quaternion based DMPs with logarithmic map (Section IV). These extensions were previously not formulated for orientation movement planing with DMPs.
- A new formulation of phase stopping coupled with standard DMPs to enable a robot to resume its movement in case of unexpected perturbations (Section IV-C). This contribution is not limited to quaternion based DMPs but applies to DMPs in general.

APPENDIX I
ROTATION MATRIX AND EXPONENTIAL MAP

The most standard way to represent rotational motion in $\mathbb{R}^3$ is through rotation matrices, which provide a nine-dimensional parametrization of $\mathrm{SO}(3)$

$$\mathrm{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3},\ \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I},\ \det(\mathbf{R}) = 1 \right\}. \quad (36)$$

Every orientation can be identified with a unique $\mathbf{R} \in \mathrm{SO}(3)$. Thus any orientation trajectory can be written as a function $\mathbf{R}(t) \in \mathrm{SO}(3),\ 0 \le t \le T$. The rotational motion of any point $\mathbf{p} \in \mathbb{R}^3$ attached to the robot's end-effector is given by $\mathbf{p}(t) = \mathbf{R}(t)\mathbf{p}_0$. The derivative of $\mathbf{p}(t)$ is equal to

$$\dot{\mathbf{p}}(t) = \dot{\mathbf{R}}(t)\mathbf{p}_0 = \dot{\mathbf{R}}(t)\mathbf{R}^{\mathrm{T}}(t)\mathbf{p}(t). \quad (37)$$

By definition of angular velocity $\boldsymbol{\omega}$ we also have

$$\dot{\mathbf{p}}(t) = \boldsymbol{\omega}(t) \times \mathbf{p}(t) = [\boldsymbol{\omega}]_\times(t)\mathbf{p}(t), \quad (38)$$

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \quad (39)$$

Note that $[\boldsymbol{\omega}]_\times$ is a skew symmetric matrix, i. e. $[\boldsymbol{\omega}]_\times^{\mathrm{T}} = -[\boldsymbol{\omega}]_\times$. Since (37) and (38) are true for any $\mathbf{p}(t)$, we obtain

$$[\boldsymbol{\omega}]_\times = \dot{\mathbf{R}}\mathbf{R}^T. \quad (40)$$

This differential equation provides the basis for the definition of orientational DMPs.

If $\boldsymbol{\omega}$ is constant, then (38) can be solved analytically

$$\mathbf{p}(t) = \exp\left(t[\boldsymbol{\omega}]_\times\right)\mathbf{p}_0 = \exp\left(\theta(t)\frac{[\boldsymbol{\omega}]_\times}{\|\boldsymbol{\omega}\|}\right)\mathbf{p}_0, \quad (41)$$

were $\theta(t) = t\|\boldsymbol{\omega}\|$ denotes the rotation angle within time t. The exponential map

$$\exp\left(t[\boldsymbol{\omega}]_\times\right) = \mathbf{I} + \sin(\theta)\frac{[\boldsymbol{\omega}]_\times}{\|\boldsymbol{\omega}\|} + (1-\cos(\theta))\frac{[\boldsymbol{\omega}]_\times^2}{\|\boldsymbol{\omega}\|^2} \quad (42)$$

is commonly referred to as Rodrigues' formula [3]. It can be shown that the net rotation $\mathbf{R}(t) = \exp\left(\theta(t)[\boldsymbol{\omega}]_\times/\|\boldsymbol{\omega}\|\right)$ is a rotation matrix, i. e. $\exp\left(t[\boldsymbol{\omega}]_\times\right) \in \mathrm{SO}(3)$. Furthermore, it can be shown that given any $\mathbf{R} \in \mathrm{SO}(3)$, there exists $\boldsymbol{\omega} \in \mathbb{R}^3$ so that $\mathbf{R} = \exp\left([\boldsymbol{\omega}]_\times\right)$ [3]. The components of $\boldsymbol{\omega}$ are called exponential coordinates of $\mathbf{R}$. This representation is often referred to as axis-angle representation of orientation. It is unique if we limit the domain of $\boldsymbol{\omega}$ to $0 \le \|\boldsymbol{\omega}\| < \pi$. Hence we can compute its inverse, i. e. the logarithmic map

$$\log(\mathbf{R}) = \begin{cases} [0,0,0,]^{\mathrm{T}}, & \mathbf{R} = \mathbf{I} \\ \boldsymbol{\omega} = \theta\mathbf{n}, & \text{otherwise} \end{cases}, \quad (43)$$

$$\theta = \arccos\left(\frac{\mathrm{trace}(\mathbf{R})-1}{2}\right),\ \mathbf{n} = \frac{1}{2\sin(\theta)}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix}.$$

To represent all orientations we need to allow also $\theta = \pi$. At $\theta = \pi$ there are always two solutions: $\pm\pi\mathbf{n}$ and $-\pi\mathbf{n}$. Numerically stable formulas to compute the exponential coordinates are given in [12]. However, there is a discontinuity in the logarithmic map at rotation matrices corresponding to rotation angles $\theta = \pi$ because at this boundary, the logarithmic map switches from positive to negative rotation angles. Since $\exp([\log(\mathbf{R})]_\times) = \mathbf{R}$, we obtain

$$\exp([\log(\mathbf{R}_2\mathbf{R}_1^{\mathrm{T}})]_\times)\mathbf{R}_1 = \mathbf{R}_2\mathbf{R}_1^{\mathrm{T}}\mathbf{R}_1 = \mathbf{R}_2. \quad (44)$$

Thus according to (41), the difference vector $\boldsymbol{\omega} = \log(\mathbf{R}_2\mathbf{R}_1^{\mathrm{T}})$ can be interpreted as the angular velocity that rotates $\mathbf{R}_1$ into $\mathbf{R}_2$ in unit time.

ACKNOWLEDGMENT

This research has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 600578, ACAT. It was also supported by MEXT KAKENHI Grant Number 23120004; by JSPS and SRA: Japan-Slovenia research Cooperative Program; by MIC-SCOPE; by JST-SICP; by SRPBS, MEXT; by contract with the Ministry of Internal Affairs and Communications entitled 'Novel and innovative R&D making use of brain structures'.

REFERENCES

[1] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computations*, vol. 25, no. 2, pp. 328–373, 2013.

[2] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control – a unifying view," *Progress in Brain Research*, vol. 165, no. 6, pp. 425–445, 2007.

[3] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, New York: CRC Press, 1994.

[4] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, California, 2011, pp. 365–371.

[5] S. Chiaverini and B. Siciliano, "The unit quaternion: A useful tool for inverse kinematics of robot manipulators," *System Analysis, Modelling and Simulation*, vol. 35, pp. 45–60, 1999.

[6] J. J. Faraway and S. B. Choe, "Modelling orientation trajectories," *Statistical Modeling*, vol. 9, no. 1, pp. 51–68, 2009.

[7] A. Ude, "Filtering in a unit quaternion space for model-based object tracking," *Robotics and Autonomous Systems*, vol. 28, no. 2-3, pp. 163–172, 1999.

[8] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, 2006.

[9] J. S.-C. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 434–440, 1988.

[10] A. J. Ijspeert, J. Nakanishi, T. Shibata, and S. Schaal, "Nonlinear dynamical systems for imitation with humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Tokyo, Japan, 2001, pp. 219–226.

[11] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "CB: A humanoid research platform for exploring neuroscience," *Advanced Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.

[12] N. Ayache, *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*. Cambridge, Mass.: MIT Press, 1991.