Auton Robot (2011) 30: 399–425
DOI 10.1007/s10514-011-9226-3

Universally manipulable body models—dual quaternion representations in layered and dynamic MMCs

Malte Schilling

Received: 2 November 2009 / Accepted: 7 March 2011 / Published online: 23 March 2011

Abstract Surprisingly complex tasks can be solved using a behaviour-based, reactive control system, i.e., a system that operates without an explicit internal representation of the environment and the own body. Nevertheless, application of internal representations has gained interest in recent years because such internal representations can be used to solve problems of perception and motor control (sensor fusion, inverse modeling) and may in addition be applied to higher cognitive functions as are the ability to plan ahead. To endow such a system with the ability to find new behavioural solutions to a given problem in a broad range of possibilities, the internal representation must be universally manipulable, i.e. the model should be able to simulate all movements that are physically possible for the body given. Using recurrent neural networks, models showing this faculty have been proposed being based on the principle of mean of multiple computation (MMC). The extension of this approach to three dimensions requires the introduction of a joint angle representation which allows for computation of mean values. Here we use dual quaternions that are singularity-free and unambiguous which allow for shortest path interpolation. In addition, it has been shown that dual quaternions are the most efficient and most compact form for representing rigid transformations. The model can easily be adapted to bodies of arbitrary geometries. The extended MMC net introduced in this article represents a holistic system that can—following the principle of pattern completion—likewise be used as an inverse model, a forward model, for sensor fusion or other, related capabilities.

Keywords Embodiment · Internal model · Mental simulation · Recurrent neural network · Dual quaternions

1 Introduction

Traditional Artificial Intelligence approaches focussed on reasoning as a process working on explicit and purely symbolic representations. Actions were only regarded as side effects of higher-level processes. Perception's only function was to inform and construct the higher-level representations. In this view, reasoning has been understood as the application of rules with action and perception being mere subordinate processes.

In contrast, during the last two decades the behaviour-based approach focussed on a completely different perspective on intelligence, or in this case intelligent behaviour: This approach does not presume representations, but concentrates on building acting systems instead, which fulfil only limited tasks like simply wandering around in an environment without running into an obstacle or, as a more complex task, acquiring information about the environment to be used for homing. The systems are assumed to be extended in a bottom-up fashion: not by imposing an overly sophisticated control structure to solve a certain task, but by evolving a controller step-by-step. It is hoped that following an evolutionary path will guide the construction of such a system in a minimalist fashion, abandoning costly extensions like unnecessary internal representations. To this end

Electronic supplementary material The online version of this article (doi:10.1007/s10514-011-9226-3) contains supplementary material, which is available to authorized users.

M. Schilling (✉)
Center of Excellence 'Cognitive Interaction Technology' (CITEC), University of Bielefeld, P.O. Box 10 01 31, 33501 Bielefeld, Germany
e-mail: mschilli@techfak.uni-bielefeld.de

M. Schilling
International Computer Science Institute (ICSI), Berkeley, USA

the environment itself can often be used instead of an internal model, an approach that is characterised by the terms *embodiment* and *situatedness* (Brooks 1991a, 1991b).

While behaviour-based systems can solve surprisingly complex tasks, e.g., starting from coordination of joints in a walking robot going up to behaviours like crossing a large gap (Brooks 1991a, 1991b; Pfeifer and Scheier 2001; Bläsing 2006; Schmitz et al. 2008), without an explicit representation of the environment, in recent years interest in internal representations has grown, because they appear to be important when such systems are to be developed further (Steels 2003). However, internal representations are now assumed to be directly influenced by and connected to motor functions or sensory influences. Higher cognitive functions use the existing sensorimotor representations instead of remodelling their function into an abstract representation (which inevitably will be error-prone).

Systems which can be called *embodied* are reactive systems in the sense that they at first exploit external representations—the environment itself. At this stage their behaviours rely on sensory data. Internal representations may only gradually co-evolve in parallel and in service for a specific action (Steels 2003). These representations may later be used for higher-level functions. As a consequence, these representations are always connected to the actions themselves and are in service for these actions (Glenberg 1997). In fact, these representations are assumed to be realised through the integration of neuronal activations in sensing and acting related to the action itself and, as a consequence, they are grounded and therefore avoid the problem of symbol grounding (Harnad 1990). To the extent that these representations concern properties of the body, we call this form of representation second-order embodiment, following Metzinger (2006), who defines it as *"generating intelligence by using an integrated representation of the body as a whole"*. As a consequence, Metzinger proposed the term first-order embodiment for what is usually termed embodiment.

The literature discusses three different tasks that rely on internal representations which in the following will be described as inverse modelling, forward modelling and models for sensor fusion.

1. Inverse modelling: If the task is to grasp a visually given object, the target position is defined in an egocentric three-dimensional space. However, the movement to reach the target must be described in terms of joint positions, displacements or muscle activations in some form. Therefore, a transformation between these two reference systems is needed. This transformation corresponds to a mapping from Cartesian space to joint space and is called an inverse model (e.g., Wolpert and Kawato 1998; Stringer and Rolls 2007). This task may become, and usually is, complicated if the body to be controlled is characterised by extra degrees of freedom (DoF), i.e., contains more joints than necessary for the solution of the task. In this case, there are not only one, but many solutions to the task (Bernstein 1967). Therefore, the controller has to select one out of these many possible solutions.

   Targeted limb movements can be found not only in humans and other "higher" animals, but already in insects, for example, in crickets (Honegger 1981), in locusts (Page et al. 2008; Matheson and Dürr 2003) and in stick insects (Cruse 1979). This suggests that inverse models may be applied by many species.
2. Forward models: When performing fast movements or adapting a movement to a fast-moving target, one can not rely solely on sensory feedback for guiding the movement, because of temporal delay inherent to the sensory and motor pathways. A possible solution is to rely on a fast prediction of the expected feedback that could be provided by a forward model (Miall et al. 1993; Desmurget and Grafton 2000). A forward model determines spatial locations when joint angles are given. Forward models are therefore often called predictors. Combined with an inverse model, a possible error can be detected faster than when relying only on proprioceptive feedback.

   When interacting in dynamic tasks, like catching a ball, it is also necessary to be able to predict the movement of target objects. Therefore a forward model that forecasts future states from the current state is required. Simple forms of predictive models can even be found in insects (for example, stick insect (Bläsing and Cruse 2004), Drosophila (Strauss and Pichler 1998), see Webb 2004 for further examples).
3. Sensor fusion: A distinct feature of animals and humans, in contrast to most technical systems, is the very high number of different modalities as well as sensors for each modality. For example, the position of an arm can be described by the visual system in a Cartesian, body-centred coordinate system while different proprioceptive sensors would provide joint angle-like representation (Makin et al. 2008). An advantage of such redundant systems is that they allow one to compensate for errors and disturbances, which however presupposes some kind of integration mechanism of the sensory information (e.g., van Beers et al. 2002; Wolpert et al. 1995; Smeets et al. 2006, integration of visual cues (Muller et al. 2009), or interpolation of positional information like in the rubber-hand illusion (Botvinick and Cohen 1998)). Relying on redundant sensory information can be found already in simple animals. An example are the different joint angle measuring sensory systems in insects, e.g. Niven et al. (2009) showed that the different sensory signals and visual information are integrated in targeted movements.

The three aforementioned tasks are examples of how basic tasks and behaviours require different sorts of internal models. These tasks establish a set of requirements which has to be fulfilled by potential implementations of such models. On the one hand, an internal model should afford the ability to represent all movements possible for a given body including its kinematic and dynamic limits (Acosta-Calderon and Hu 2005). On the other hand, it must be able to relate different modalities to account for the diversity and redundancy of sensors in natural systems (de Vignemont 2010). In other words, such a body model should be universally adaptable for perceptions and universally manipulable when it is used for motor control and imagining body movements.

At the same time, the different functions of internal models show that even quite basic behaviours as, e.g., grasping or reaching, rely on quite complex internal models and representations of complex structures, at first of the own body but later-on of objects and the environment. This opens up an evolutionary path on which internal models may have co-evolved in service for specific actions as demanded by Steels (2003). When these internal models became sufficiently complex, they may be used in a different context and for a different purpose. An internal model being capable of the forward function could be used to predict the effect of a behaviour. When the model can be decoupled from the body (Hesslow 2002; Wilson 2002), the internal model may be used as a simulator for planning ahead. The internal model can then be used to try out new behaviours in a mental simulation or as done in imagination without actually performing them (Jeannerod 1999). If this ability to predict is complemented by the ability to use these predictions as a basis to decide on what behaviour will be performed, the system is not anymore a reactive system, but can, according to the definition of McFarland and Bösser (1993), be termed a cognitive system. Therefore, if evolution has equipped a brain of a reactive system with such a network, the step to exploit the predictive capabilities of this network and thereby to become a cognitive system appears to be a small one. The mechanism of internal simulation is critical for this notion of a cognitive system, i.e., a system which is able to plan ahead. Internal simulations means to imagine the consequences of its possible actions and to choose an action which maximises the benefit for the system—or as Shaw (1903) has put it *"To be able to choose the line of greatest advantage instead of yielding in the direction of least resistance"*.

Traditional solutions in robotics apply separate monolithic models for each of the three tasks above, all of them involving internal models. Wolpert and Kawato (1998) offer a way of dealing with the inverse and the forward task by introducing for each behaviour a pair of dedicated inverse and forward models. The MOSAIC architecture (Wolpert and Kawato 1998) has shown its ability to control systems with a set of simple behaviours and to choose the appropriate behaviour. Instead of applying a pair of separate models, Morasso and Sanguineti (1994) proposed a body model by combining both models in such a way that the output of the inverse model is connected with the input of the forward model and the output of the forward model with the input of the inverse model. Thus, the body model is represented by a recurrent network.

When addressing the third function, sensor fusion, application of Kalman filtering has been proposed (Wolpert et al. 1995). Although this method can be applied to systems with redundant degrees of freedom, due to a minimisation procedure required for inverse modelling, only specific solutions can be provided (Grush 2004). In other words, such a system cannot exploit all possibilities the system is capable of realising when exploiting its extra degrees of freedom.

In contrast, Cruse and Steinkühler (1993) proposed a holistic network, forming a unified inverse-forward model, which can represent any configuration that is geometrically possible and may therefore be termed a universally manipulable body model. While this network, termed mean of multiple computation (MMC) network, can be used as a forward model, as an inverse model and as any mixed model (Cruse and Steinkühler 1993; Steinkühler and Cruse 1998), it was originally applied only to 2D structures and the extension to 3D structures was restricted to Cartesian coordinates. An extension to general coordinates including joint angles, for example, requires additional processing of constraints which introduce non-linearities. In this article the MMC principle is now applied to rigid transformations in general which include translations and rotations in three dimensions. The extension of angular relationships to three-dimensional geometries is not trivial. We apply the approach of dual quaternions that can be realised by using non-linear neural networks and leads to a more elegant structure. Therefore, the MMC principle will be briefly reviewed first, followed by an introduction to the dual quaternion structure.

The MMC network represents a pattern-completion system that can be used for any task, forward modelling, inverse modelling, any mixed case or for perception/sensor fusion, depending only on the selection of the input values. As an example structure we use a three-segmented arm performing a reaching task. In such an inverse kinematic problem, a target position described in a three dimensional Cartesian space is given to the network. One task for the network is to come up with a solution of how the joints of the manipulator, e.g., a human-like arm, have to be moved to reach the target position. But the network could further be used in different contexts for sensor fusion, e.g. to cancel out noise or errors, or for computation of the forward kinematics.

This article addresses the structure of the network and shows how the network can be used to solve inverse kinematic problems. Results are presented and are, in the last

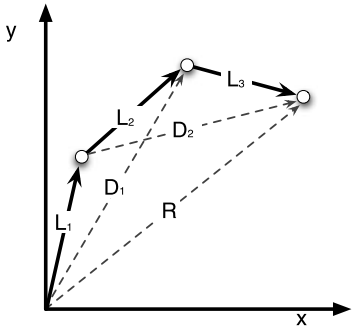

Fig. 1 Graphic representation of a planar (2D) arm consisting of three segments, upper arm (L_1), lower arm (L_2) and hand (L_3). Vector R points to the position of the end effector (tip of the hand). D_1 and D_2 represent additional diagonal vectors

section, discussed and related to other approaches. The usage for sensor fusion and prediction will only briefly be discussed as well as how the network can be incorporated into a controller scheme as such (for a detailed explanation on how such an internal model may be used for control and planning ahead in a six-legged walking robot see Schilling and Cruse 2008).

2 MMC—an internal model

The MMC model is implemented as a neural network and fulfils the aforementioned requirements, i.e., it can be used as an inverse model, a forward model or a model for sensory integration.

At first, we will briefly review the general idea—the mean of multiple computation (MMC) principle—and how to construct a neural network model for a very simplistic kinematic example. Following the introduction of dual quaternions as a suitable representation of the kinematics of a body, the principle will be applied to a 3D body model.

2.1 Classical MMC approach

The classical MMC network (Cruse and Steinkühler 1993; Steinkühler and Cruse 1998) is an autoassociator, a type of self-organising map that is constrained by geometric relationships representing the kinematics of, to take a simple example, a serial chain manipulator, or, as a more complex case, of a six-legged walker with 18 degrees of freedom. By means of the geometric constraints, the network is forced into its attractor states, which always represent valid, geometrically correct solutions. In the following, we will use a three-segmented planar arm as a simple example.

Figure 1 shows the robot arm: the arm, which is restricted to movements in a plane, consists of three segments that are connected by two hinge joints forming a serial chain and attached to a fixed point by a third hinge joint (the "shoulder joint"). Following the mean of multiple computation approach, each variable is described only by relative relations of the adjacent variables. However, each variable may be part of many equations (therefore allowing for multiple computations).

The simple 2D, three degrees of freedom manipulator shown in Fig. 1 is represented by the vectors characterising its segments. In addition, we introduce diagonal vectors in such a way that each node of the manipulator—these are the joints and the end-effector—is connected with each other node. Local equations can be formulated that describe the geometric relationships among all these vectors. A local relationship simply represents a triangle of three vectors. When we construct all possible triangles, we obtain $\binom{n}{3}$ equations. In our example we have 4 triangles and each variable is part of two triangles:

$$\begin{aligned} L_1 + D_2 - R &= 0 \\ L_1 + L_2 - D_1 &= 0 \\ D_1 + L_3 - R &= 0 \\ L_2 + L_3 - D_2 &= 0 \end{aligned} \tag{1}$$

Next, for each variable we solve the two equations of which they are part of. Now every variable is described by a set of—in our simple example two—equations.

$$\begin{aligned} L_1 &= R - D_2 \\ L_1 &= D_1 - L_2 \end{aligned} \tag{2}$$

The general idea of the mean of multiple computation principle is to compute the multiple equations for a variable in parallel and to assign the mean value of these multiple computations to the variable itself. The whole process is iterative. As a consequence, the value of a variable can change over time. The new value for a variable is calculated as described above and shown for the first segment as an example:

$$L_1(t+1) = \frac{1}{2}(R(t) - D_2(t)) + \frac{1}{2}(D_1(t) - L_2(t)) \tag{3}$$

The resulting equations can now be written as a weight matrix describing a recurrent neural network (see Fig. 2). A prerequisite is to decompose the vectors into their components. For every single component (here x- and y-components), a unique network has to be built. In our example, the two resulting weight matrices are identical. In the simplest case, the diagonal of the weight matrix consists of zeros. To prevent oscillations it has shown to be sufficient to introduce a damping factor d in the diagonal of the weight matrix (discussed in the Appendix of Steinkühler and Cruse 1998).

$$\begin{aligned} L_1(t+1) = {} & \frac{1}{d}(R(t) - D_2(t)) + \frac{1}{d}(D_1(t) - L_2(t)) \\ & + \frac{d-2}{d} L_1(t) \end{aligned} \tag{4}$$

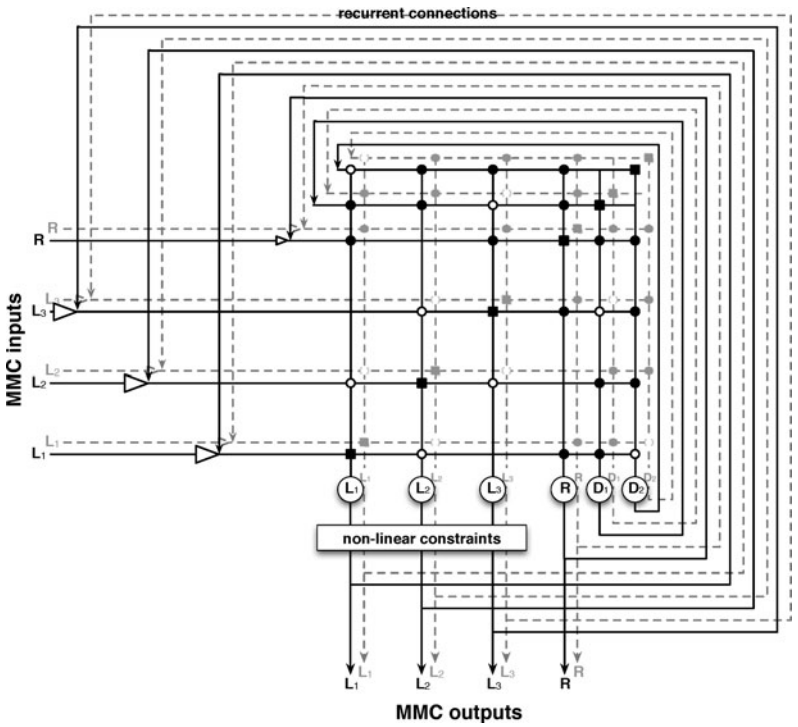

Fig. 2 A recurrent neural network containing 2×6 units. The complete net consists of two identical linear networks, one for the x-components (*black lines*) and the other for the y-components (*grey dashed lines*) of the vectors. The units represent the components of the six vectors L_1, L_2, L_3, D_1, D_2 and R of the planar arm (see Fig. 1 for graphic illustration). If an input is given, the corresponding recurrent channel is suppressed (symbolised by the *open arrow heads*). For details see text

The damping values can be related to the time constant of a low-pass filter (Makarov et al. 2008). They can differentially influence the relaxation dynamics, but do not affect the final position.

When the network is in a stable state, the different equations describing one variable lead to exactly the same value—the geometrically correct solution. The different equations are redundant. As a consequence, when we introduce a disturbance into the network by changing one variable, the network starts to compensate for the disturbance and approaches a new state in which the disturbance is distributed over all the equations and the error is minimised. In other words, the network compensates for this error and relaxes to a stable state in such a way that it adjusts its internal values until all the equations are satisfied again and the network represents a correct solution (see Fig. 3 for an example). When setting the variable describing the tip of the end-effector—R, see in Fig. 1—to a new value, the equations for the calculation of the other variables containing the R vector are influenced correspondingly. During all iterations, the new target value is imposed upon the network, therefore overriding the calculation of new target vectors (in Fig. 2 this is represented by the open arrow heads). As a result, the network relaxes to a solution that maintains the value of R. Thus, the network solves the inverse kinematic problem. In Fig. 3(a) the state of the arm is shown for different points of time and in (b) the velocity of the end effector is plotted.

The function of the network can be thought of as the application of the passive motion paradigm as stated by Mussa-Ivaldi et al. (1988) to solve the inverse kinematic problem for surplus degrees of freedom. The network behaves as if we had a real stick model of the manipulator and now pulled this model at its tip with a rubber band to the target location. The tip will move towards the target while the segments and joints will just follow the movement. In principle, the MMC network does the same. While "pulling" at the tip by changing vector R, all equations are pulled in a direction to counterbalance the movement at the tip and to distribute the movement equally to all variables.

However, as explained, until now, there remains one major problem: In the MMC network all the constraints are treated equally. When changing the x- and y-components, the joint angles as well as the segments length are adapted. However, unless systems with prismatic joints were used, changing the length of a segment is generally not intended. In the classical MMC approach, additional external constraints have been introduced that counteract the segment-length change. While the convergence of the model is guaranteed for the linear MMC model (Steinkühler and Cruse 1998), i.e., a model without external constraints acting on some variables, this has not be proven in general for the model extended through constraints which act in a non-linear way on the variables. Until now there are only proofs for special cases (Steinkühler 1994).

To apply constraints, it was in general necessary to transform the variables in the network into a different representation form (Schilling and Cruse 2007)—scaling the segments in Cartesian Coordinates can be thought of as transposing the coordinates to a representation of a direction and a length

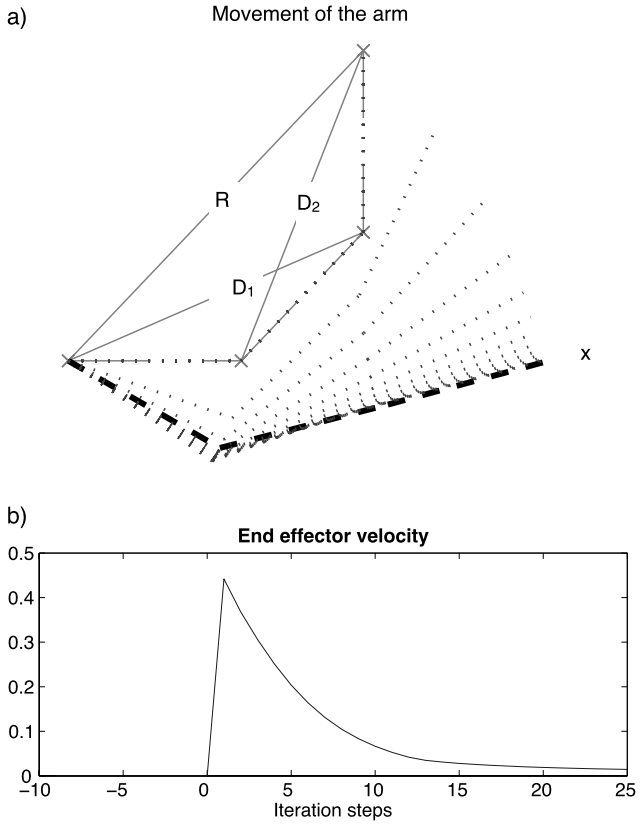

Fig. 3 Solution of the inverse kinematic problem. A planar arm with three segments (i.e., one extra DoF) should point to a given position, marked by a *cross*, starting from an initial configuration. In (**a**) the state of the arm for every second iteration step is shown. In (**b**) the velocity profile of the end effector is plotted over time

of the vector. A more elegant way is to use dual quaternions to represent positions, rotations and translations which allows us to use one single type of representation and make explicit transformation unnecessary. As a consequence, it is easy to maintain constant segment lengths. In addition, while applying the passive motion paradigm does not circumvent that the arm may be pulled near to its joint limits, the type of representation chosen here allows us to easily introduce external constraints. In this way approaching joint limits can be avoided on a local level by steering each single joint away from its workspace limits. Such extensions, as have been proposed by, e.g., Yoshikawa (1985) or Cruse (1986), Cruse and Brüwer (1987) can be easily applied to the MMC model as proposed here.

2.2 Dual quaternion MMCs

Above, we introduced the mean of multiple computation principle using a vector-based model that describes a manipulator in two-dimensional Cartesian space. Usually, such manipulators are represented in a different format and a configuration is given by the position of the joints. The latter representation is better suited for control and for planning movements because controllers usually have to deal with the task of transforming a problem description given in a global 3D space into a joint configuration. While such joint representations and transformations are trivial for two-dimensional spaces, making the application of the MMC principle for this case straightforward, the extension of trigonometric relations to three dimensions is complicated.

Usually homogenous coordinates are used to represent affine transformations. Homogenous transformation matrices (Maxwell 1951) are 4×4 matrices forming the group of rigid body displacements, called the Special Euclidean group $SE(3)$ (Murray et al. 1994). The advantage of homogenous transformation matrices is that translational and rotational parts are represented at distinct places in the matrixand that the concatenation of two transformations equals the multiplication of the two matrices. Translations can be easily described by a relative position, that is a three-dimensional vector. Rotations in two dimensions are easily represented through an angle of rotation and a rotation centre that is a point in space. In combination with translations, the origin is usually chosen as the rotation centre because the computations of the rotation are then trivial. A description of a rotation in three dimensions is more difficult. Often, an orientation is described by means of a concatenation of three standard rotations, which are rotations around one of the three axes of the coordinate system in a predefined order. The orientation is represented by means of the three angles of this standard rotations, called Euler angles. One set of such standard rotations is given by the Tait-Bryan rotations, i.e., the Yaw, Pitch and Roll angles. The representation through homogenous transformation matrices and especially the use of Euler angles for representing orientations has some serious disadvantages (Aspragathos and Dimitros 1998; Wang 1999; Klein Breteler and Meulenbroek 2006):

– Singularities: Euler angles form a chart with the special orthogonal group of rotations in three dimensional space. This chart is mostly smooth, but there are singularities: The so called “Gimbal lock”, occurring when, following a rotation, two of the rotation axes are aligned in parallel, so that one degree of freedom is lost. Such a singularity is characterised by the fact that small changes in one representation may lead to very large changes in the other (Wang 1999; Klein Breteler and Meulenbroek 2006). In this case, infinitesimal changes of orientation may result in fast and huge changes in the Euler angles, which may cause movements with high velocity.
– Ambiguity: Another problem with the Euler angle representation is that a given orientation can be represented by an infinite number of different combinations of Euler angles (Foley et al. 1996). The map from Euler angles to the Special Euclidean group is only surjective and not bijective. This is a serious disadvantage with respect to using

such a representation for calculating mean values as required by the MMC principle. In such a case, the result of the mean calculation of the involved parametrisations depends not only on the orientations themselves which should be averaged. Instead, the result also depends on which set of possible Euler angles have been chosen to represent the orientation.
- Ambiguous interpolation paths: As a consequence, the result of an interpolation can not be guaranteed to follow the shortest path (this extends to homogenous transformation matrices in general).
- Normalisation required in interpolation: Interpolation between two homogenous transformation matrices in general is problematic. There is no simple computation of a mean transformation or a possibility to combine two weighted transformations. The resulting matrix does generally not describe a valid transformation. A normalisation of the matrix is required, which is quite costly.
- Compactness and efficiency: Orientation and position can be represented by six numbers, but homogenous transformation matrices use 16 numbers. Therefore, there is a high degree of unnecessary redundancy. In addition, the concatenation of two transformations becomes overly expensive (Funda and Paul 1990; Aspragathos and Dimitros 1998).

The main problems are related to the representation of rotations: First, when describing an orientation by a set of standardised rotations the orientation can be mapped on multiple sets of rotations. Second, choosing a 3×3 rotation matrix as part of the homogeneous transformation matrix to represent the rotation is inefficient. Ideally, a representation of a rotation should describe a rotation in a compact way, allowing for efficient computation. In addition, there should be no singularities in the representation. Concatenation as well as interpolation between transformations should be easy. A suitable description of orientations or rotations is possible by choosing an axis-angle notation: The rotation is described by the rotational axis and by the angle giving the amount of rotation.

Quaternions provide a compact and efficient way of representing an axis-angle rotation which at the same time circumvents the aforementioned problems and can be easily used for shortest path interpolation (see Appendix A, and Hamilton 1844, 1866; McCarthy 1990; Bottema and Roth 1979): quaternions are quadruples ($\mathbf{q} = w + xi + yj + zk$), forming a normed division algebra over the real numbers with three imaginary units (i, j, k). They can be thought of as an extension of the complex numbers, but, while addition and multiplication are well-defined for quaternions as in complex numbers, the commutativity is lost, thus forming a skew field.

Quaternions are well suited to represent rotations. On the one hand, the set of all three dimensional rotations is a three-dimensional manifold whose topology is not trivial. It is known as the special orthogonal group in three dimensions or the rotation group for three-dimensional space $SO(3)$. On the other hand, all unit quaternions form a unit three-sphere S^3 and form a group under multiplication. There is a direct relation between both groups. Unit quaternions capture the topology and structure of the manifold of the rotation group. The group of unit quaternions is a double cover of the group of rotations in three-dimensional space. This means that every rotation corresponds to two unit quaternions ($\mathbf{q}$ and $-\mathbf{q}$). While this twofold redundancy is important in some specific application, e.g., for the description of spin in quantum mechanics, it can be neglected in our case. The combination of two rotations can be realised as the multiplication of the corresponding unit quaternions. As such, unit quaternions provide a compact, stable and efficient way to express rotations and use these in computations. Therefore, they have become a standard tool for expressing rotations in Computer Graphics (Shoemake 1985). There, quaternions are used to calculate combinations of rotations (and reflections) and for interpolation. But for the calculation of transformations in general other representation formalisms are used which can deal also with translations. This makes explicit conversions before and after the use of quaternions necessary which is quite inefficient and therefore not desired. To avoid these conversions, one would need an extension of the quaternion representation in which the properties of the quaternions are still given and which can be used to represent translations.

How can translations be represented? One possible extension of the quaternions is the dual quaternions (Clifford 1882; Kavan et al. 2008; McCarthy 1990; Bottema and Roth 1979). Dual quaternions do not consist of real numbers, but instead use dual numbers. Dual numbers extend the real numbers through the introduction of the unit ε—for which $\varepsilon^2 = 0$ holds true (with ε nilpotent)—constructing a two-dimensional commutative associative algebra. A dual number $\hat{a}$ consists of a non-dual part a_0 and a dual part a_ε ($\hat{a} = a_0 + \varepsilon a_\varepsilon$). As a consequence, a dual quaternion consists of eight numbers. Like the quaternions, the dual quaternions form a non-commutative algebra over the reals. The real part of the unit dual quaternions (q_0) consists of four values and represents rotations in the same way as a unit quaternion. The important extension is the introduction of the dual part, which can be used for representing translations. A translation by a vector about (t_x, t_y, t_z) can be written as the unit dual quaternion $\hat{\mathbf{q}} = 1 + \frac{\varepsilon}{2}(t_x i + t_y j + t_z k)$ (the real vector part of the dual quaternion equals zero). In analogy to using half-angles for the representation of rotations (see Appendix A), we also use half-translations. Then the composition of transformation can be defined in exactly the same way as for quaternions: for applying a transformation (represented by the unit dual quaternion $\hat{\mathbf{q}}$) to a position P (written as the dual quaternion $\hat{\mathbf{p}} = (v_x i + v_y j + v_z k)$) one has to compute

$$\hat{\mathbf{q}}\hat{\mathbf{p}}\overline{\hat{\mathbf{q}}^*} \tag{5}$$

$\hat{\mathbf{q}}$ can represent any transformation, a rotation, a translation, a combination of both that can be accomplished by dual quaternion multiplication (the order is critical) or any sequence of transformations. $\overline{\hat{\mathbf{q}}}^*$ is the conjugate of the dual quaternion $\hat{\mathbf{q}}$ (see Appendix A). In this way, unit dual quaternions can be used to represent the transformations describing kinematic relations. The multiplication of two such dual quaternions corresponds to the concatenation of two transformations. As for quaternions, the dual quaternions can be geometrically interpreted: a quaternion can be written as $\mathbf{q} = \cos\frac{\theta}{2} + \mathbf{s}\sin\frac{\theta}{2}$. The scalar part expresses the angle of the rotation and the vector part expresses the normalised axis of rotation ($\mathbf{s}$ is a unit quaternion describing a vector, i.e., the scalar part is zero). In a similar way, a dual quaternion can be written as $\hat{\mathbf{q}} = \cos\frac{\hat{\theta}}{2} + \hat{\mathbf{s}}\sin\frac{\hat{\theta}}{2}$ with $\hat{\theta} = \theta_0 + \varepsilon\theta_\varepsilon$, $\hat{\mathbf{s}} = \mathbf{s}_0 + \varepsilon\mathbf{s}_\varepsilon$. It can be interpreted as the representation of a screw motion (Blohm and Crawford 2007). Following Chasle's theorem (Daniilidis 1999; Chasles 1830), every rigid transformation can be described as a rotation around an axis and a translation along the same axis. The real part of the dual quaternion is used to represent the rotation. θ_0 is the rotation angle and $\mathbf{s}_0$ the rotation axis. The translation along the rotation axis is given by θ_ε. $\mathbf{s}_\varepsilon$ represents the position in space of the axis in an invariant way (this is called the *moment*, see Kavan et al. 2008).

Dual quaternions have been used to describe rigid transformations in robotics (Yang and Freudenstein 1964; McCarthy 1990; Bottema and Roth 1979), computer graphics (Kavan et al. 2007, 2008) and other fields. They offer a simple way to describe arbitrary geometric relations and provide a mathematical background that allows combining transformations by simple multiplication of the corresponding dual quaternions. In a comparison of different representation formalisms, Funda and Paul (1990) have shown that dual quaternions are the most computationally efficient and most compact representation. For application in MMC networks, it is essential to be able to calculate a mean value of two individuals representing transformations, i.e., to be able to interpolate between the two (see Appendix B). In this respect, dual quaternions are robust, exhibiting no singularities and being unambiguous (besides the antipodal property, which is not relevant for our case). Quaternions—and dual quaternions even more so—are often abandoned because they, as 4-dimensional structures, are difficult to imagine or illustrate. However, not only do they provide a geometrically illustrative interpretation, one also gains a lot from the mathematical foundation of the quaternions: the extension to four dimensions is essential for the properties, like the topology of the manifold.

In the following, we use dual quaternions for the construction of an internal model. The kinematics will be represented by dual quaternions and the MMC principle applied to solve the kinematic tasks. The geometric structure will be described by translations and rotations:

- Segments will be described by constant translations along the segment (of course, constant changes of orientations could be included here as well) being termed $\hat{\mathbf{t}}_i$.
- Joints are represented as variable rotations or translations. In hinge joints, the joint axis is fixed, while in ball joints this axis can also be changed and a prismatic joint can be represented by a variable translation. Below, we will concentrate on rotational joints, which will be named $\hat{\mathbf{r}}_{\theta_i}$.
- Additional diagonal vectors correspond to translations and a set of rotations aligning these diagonals with the coordinate frames given by the robot structure. These are given as $\hat{\mathbf{t}}_{d_i}$ and $\hat{\mathbf{r}}_{\delta_i}$, or $\hat{\mathbf{r}}_{\gamma_i}$.
- The end effector position and orientation can be described in the same way by a translation and a set of rotations given as $\hat{\mathbf{t}}_r$ and $\hat{\mathbf{r}}_\alpha$, or $\hat{\mathbf{r}}_{\delta_i}$.

2.2.1 *Construction of the multiple equations—describing the forward kinematics*

Following the MMC principle, the kinematics should be described in multiple ways. For the illustration, we will again use the three-segmented arm (see Fig. 4). As a starting point, the geometric structure is divided into subparts in exactly the same way as has been described above for the classical approach: triangles are used to describe the local geometrical relationship, reducing the complexity and simplifying the generally possible three-dimensional geometrical structure to relations acting in one plane. In our example, we end up with the same set of triangles as in the planar approach, but the representation is different (see Fig. 4): a kinematic chain is described by the concatenation of transformations. Some of these transformations are kept fixed (the segment describing translations), while others are variable. The variable ones are, on the one hand, the rotations describing the joints. On the other hand, the variables represent the diagonal and end effector transformations with variable lengths, which result from the forward kinematics.

As an example of setting up the equations of the MMC network, we will concentrate on the description of the first diagonal D_1 (see Fig. 5), given by a rotation and a translation starting from the origin. We equate the generating transformation for the given configuration of the arm which is called a motor (a motor includes rotational and translational information): The transformation to get from the origin to the position of the second joint can be reached by the concatenation of the rotation $\hat{\mathbf{r}}_{\gamma_1}$ followed by the translation along this diagonal $\hat{\mathbf{t}}_{d_1}$.[1]

[1]For calculation of the position with respect to the origin, following (5) we have to apply this transformation to a dual quaternion describing the origin in the origin's reference frame: $\hat{\mathbf{r}}_{\gamma_1}\hat{\mathbf{t}}_{d_1}\hat{\mathbf{p}}(\hat{\mathbf{r}}_{\gamma_1}\hat{\mathbf{t}}_{d_1})^*$ (note the right multiplication of the following translation, as the translation is with respect to the rotated coordinate frame). In the following, we

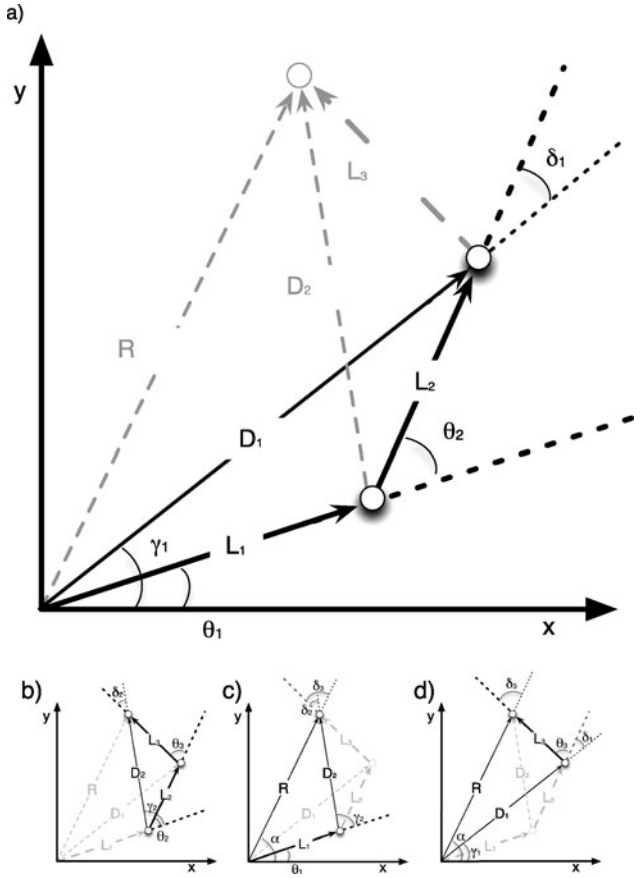

Fig. 4 Graphical representation of the three-segmented arm, upper arm (L_1), lower arm (L_2) and hand (L_3). Vector R points to the position of the end effector (tip of the hand). D_1 and D_2 represent additional diagonal vectors. The figures illustrate how the equations can be constructed. Each equation describes a triangle. The sides of the triangle are the segments of the manipulator, the diagonal vectors and the vector to the position of the end effector. These vectors are described through a rotational part around the starting point of the vector and through a translation following this rotation. Figure (**a**) to (**d**) show the four different possible triangles and introduce segment names and angle names

In the same way—through the concatenation of transforma tions—the generating transformation can be calculated through the forward kinematics of the robot structure: $\hat{\mathbf{r}}_{\theta_1}\hat{\mathbf{t}}_{l_1}\hat{\mathbf{r}}_{\theta_2}\hat{\mathbf{t}}_{l_2}$. These two transformations describe the same configuration of the manipulator. In addition, we have to introduce an auxiliary variable describing the necessary rotation to align the two orientations in the second joint: $\hat{\mathbf{r}}_{\delta_1}$. Following this operation, the diagonal can be described as: $\hat{\mathbf{r}}_{\gamma_1}\hat{\mathbf{t}}_{d_1}\hat{\mathbf{r}}_{\delta_1} = \hat{\mathbf{r}}_{\theta_1}\hat{\mathbf{t}}_{l_1}\hat{\mathbf{r}}_{\theta_2}\hat{\mathbf{t}}_{l_2}$. This equation can now be solved for each dual quaternion describing a variable transformation. As an example, we can obtain $\hat{\mathbf{r}}_{\theta_1}$ which equals the computation of L_1 in (2) in the classical MMC approach:

$$\begin{aligned}\hat{\mathbf{r}}_{\theta_1} &= \hat{\mathbf{r}}_{\gamma_1}\hat{\mathbf{t}}_{d_1}\hat{\mathbf{r}}_{\delta_1}\left(\hat{\mathbf{t}}_{l_1}\hat{\mathbf{r}}_{\theta_2}\hat{\mathbf{t}}_{l_2}\right)^{-1} \\ &= \hat{\mathbf{r}}_{\gamma_1}\hat{\mathbf{t}}_{d_1}\hat{\mathbf{r}}_{\delta_1}\hat{\mathbf{t}}_{l_2}^{-1}\,\hat{\mathbf{r}}_{\theta_2}^{-1}\,\hat{\mathbf{t}}_{l_1}^{-1}\end{aligned} \tag{6}$$

As for the classical MMC approach and according to the MMC principle, for every variable, different multiple computations are generated by using different interrelations between the triangles (see Sect. 3, in the electronic supplementary material for a complete list of the equations). In the example, for each variable, two different computations are obtained. The second one describing the rotation of the first joint involves the end effector position and the second diagonal vector.

Following this approach, the equations can be set up describing the kinematic chain. The network can directly compute the forward kinematics when the joint angles, i.e., the position of the end effector, are given. When dynamic equations are included in the computation of the mean, the movement of the arm to the end configuration can be predicted and simulated. For the computation of the inverse kinematic, it is necessary to introduce for each joint an error compensation, because when enforcing a new end position the network compensates for this disturbance by distributing the disturbance onto all parameters of the free variables. As a consequence, translational shares arise in the dual quaternions representing the joints. This effect can be balanced by transforming the translational error into a compensating rotation of the joint (see Appendix C).

Critical for the mean of multiple computation approach is, besides the parallel multiple computation of equations, the calculation of a mean value. For rigid transformations, it is not quite obvious how to construct a mean transformation as a result of a set of transformations. Ambiguous representation formalisms can not lead to an unambiguous calculation of a mean transformation. Therefore such ambiguous representations are not suited for our approach. Angle-axis representations for rotations and screw motions for representing transformations in general are not afflicted with this problem. There are methods for interpolating between quaternions that have recently been expanded to dual quaternions as well, allowing the calculation of a mean for a set of transformations (see Appendix B for details on different methods that produce exact solutions or a good approximation following a simple principle). We used dual quaternion linear blending (see Appendix) for computation of a weighted mean of dual quaternions as it is a simple and efficient method.

2.3 MMCs containing dynamic equations

One problem with classical MMC networks is that the network is approaching the solution with a velocity profile that

are concentrating only on the transformations and, due to the specific properties of dual quaternions with respect to ambiguity, it is allowed to equalise transformations obtained following different paths.

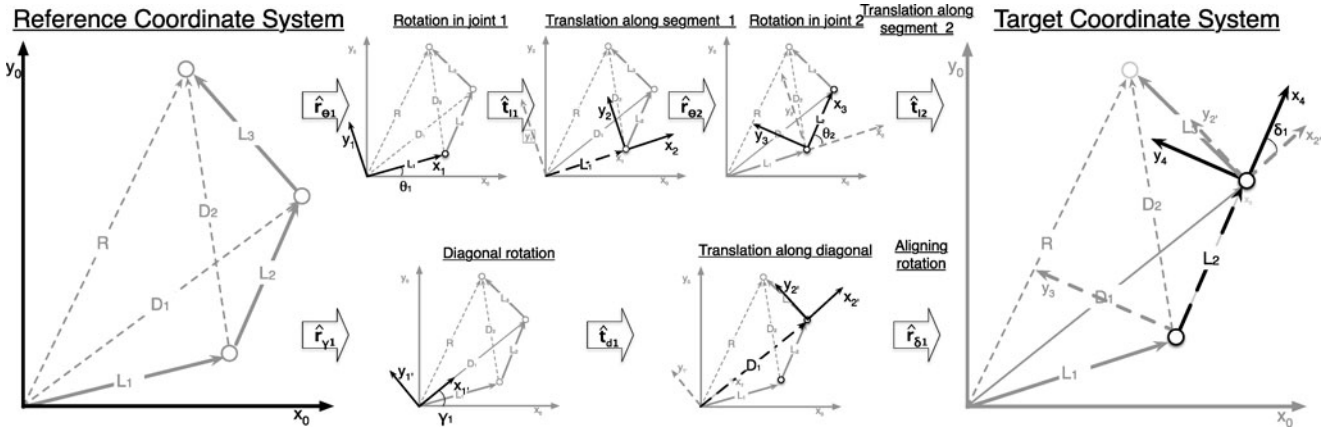

Fig. 5 Graphical representation of the three-segmented arm, upper arm (L_1), lower arm (L_2) and hand (L_3). Vector R points to the position of the end effector (tip of the hand). D_1 and D_2 represent additional diagonal vectors

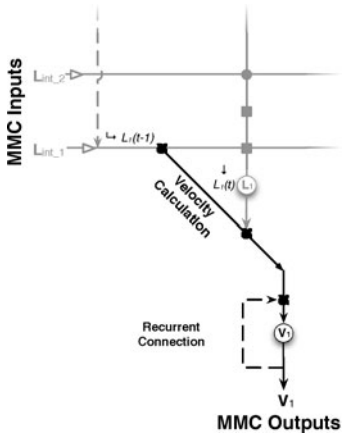

Fig. 6 Calculation of the velocity in the MMC network shown in Fig. 2. For details see text

starts with a high velocity, but later-on decreases exponentially. Changing the damping factors can only influence the time constant, but not the characteristic velocity profile. Human arm movements are characterised by very different velocity profiles usually being bell-shaped (Morasso 1981) for small movements. Is it possible to introduce dynamic influences into the model leading to such bell-shaped profiles? In the following we show how the MMC principle can be exploited to specifically control the relaxation velocity.

Velocities are defined as position changes over time ($v = \frac{\delta s}{\delta t}$). The MMC network lacks any explicit representation of time. Instead, changes of positions are linked directly to the network dynamics and the iterative process of approaching an attractor is time dependent. This process depends on the iteration steps. As a consequence, the velocity can be written for the classical vector MMC as $v(t) = \frac{s(t)-s(t-1)}{\delta t*(t-(t-1))} = \frac{1}{\delta t} * s(t) - s(t-1)$. $s(t)$ is the output value of the network at time t. During the relaxation process, the input variables are determined by the recurrent connections, i.e. the input values are the variable values from one time step ahead at $(t-1)$. The neural network therefore includes both values needed: the current value of a variable and as an input the preceding value of this variable. The difference between these two values in the vector MMC represents the change of the variable during one time step being proportional to the velocity. The velocity can be illustrated as a diagonal connection in the neuronal network which calculates the difference between input and output (see Fig. 6). The equations describing the velocities can now be included as one of the multiple computations like any other equation. Calculation of the mean can be used to integrate multiple velocity equations (or even more to connect these equations describing accelerations).

Applied to the dual quaternion representation the change of a joint variable during a timestep represents a rotation. This rotation represents the angular velocity of the joint. As rotations are concatenated by multiplication, the joint velocity is not computed as the difference between the joint angle at time t and the joint angle at time $t-1$, but it uses the quotient:

$$\hat{\mathbf{r}}_{\omega_i}(\mathbf{t}) = \hat{\mathbf{r}}_{\theta_i}(\mathbf{t}-\mathbf{1})^{-1} * \hat{\mathbf{r}}_{\theta_i}(\mathbf{t}) \tag{7}$$

ω_i corresponds to mapping the old joint value onto the current joint value.

To illustrate how this extension allows to access the dynamic variables in a flexible manner, we will apply the extension to our earlier example, the three-segmented arm. We will add three joint velocity variables and the corresponding equations. In this simple extension, we are not using multiple velocity equations for each joint (like equations relating velocity to inertia, momentum or energy). Instead, to make the example as simple as possible, we are only extending the neuronal network in such a way that the dynamics of the arm in forward and inverse dynamic problems can be represented and accessed. No additional influences have to be in-

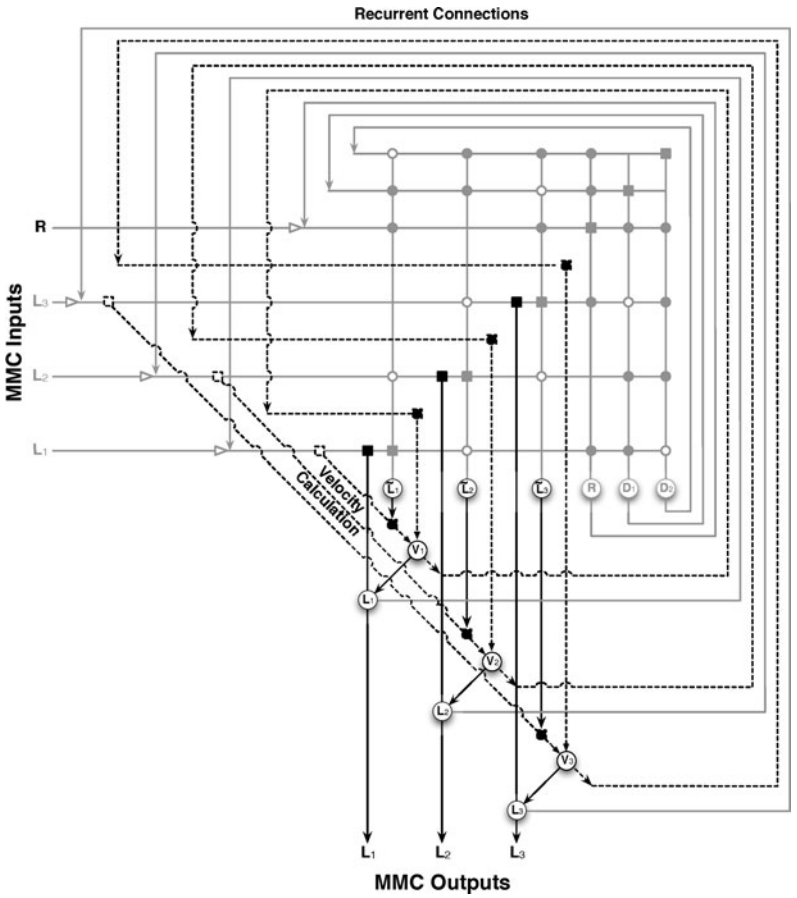

Fig. 7 The recurrent neural network containing the velocity equations. The complete net consists of two identical linear networks (application of non-linear constraints not shown), one for the x-components and the other for the y-components (not shown, but identical to the network representing the x-component) of the vectors. The units represent the components of the six vectors L_1, L_2, L_3, D_1, D_2 and R of the planar arm (see Fig. 1 for graphical illustration). If an input is given, the corresponding recurrent channel is suppressed (symbolised by the *open arrow heads*). The velocities are calculated as the difference between two time steps. The output position for the segments is calculated by applying the internally calculated offset given by the velocity to the preceding position. For details see text

tegrated in this rather simplistic case, only the recurrent connections for the velocities are fed back to the network and weighted by a damping factor—the velocity damping factor d_{vel}. The introduction of the recurrent connections allows us to influence how strongly the new computed velocity—that is used as a control signal—should depend on the current velocity by changing the velocity damping factor d_{vel}. A high damping value would lead to a smooth velocity profile with low accelerations while a low damping value would allow to quickly accelerate. This small recurrent network shows low-pass properties and provides a form of a dynamic model for the movement of the arm (the velocity damping factor d_{vel} can be thought of as being related to the inertia of the connected segments). This model is of course an oversimplification as it reduces the overall dynamic influences to low-pass properties. But while its simplicity allows to explain how dynamic influences can be integrated, the network is sufficient to produce quite naturalistic arm movements.

As indicated in Fig. 6 for a classical vector MMC network the kinematic network operates as an internal part of the whole network including velocity equations, producing a new internal position estimate ($L_{int\_i}(t)$). This is used for the calculation of the velocity. But the overall output of the network including the velocity influences can be computed as the sum of the current positional information ($L_{int\_i}(t-1) = L_{output\_i}(t-1)$) and the positional change as given by the computed velocity (see in Fig. 7 the whole network). This transfers to the dual quaternion network, as the joint angles derived from the kinematic equations are forming the internal network (θ_{int_i}) which are used for calculating the joint angle change during one time step (see ω_i, (7)). The control signal ($\theta_{output\_i}(t)$) is computed as the joint change $\omega_i(t)$ applied to the old joint position $\theta_{int\_i}(t-1) = \theta_{output\_i}(t-1)$. The change of the joint angles therefore does not solely depend on the neural network dynamics of the geometry describing part of the MMC, but is coupled to the velocity and can be linked to other velocity influences (for further details see Schilling 2009).

The advantage of this network structure is that we are now able to alter the velocity-dependency of movements directly and introduce influences on the velocity into the network. This network can be used in a flexible way for different purposes by adjusting the two damping factors.

- It can still be used for any forward, inverse or mixed kinematic problem in the same way as the kinematic MMC network through turning off the velocity damping factor ($d_{vel} = 0$).
- On the other extreme, when only using the velocity related part of the network and switching the recurrent con-

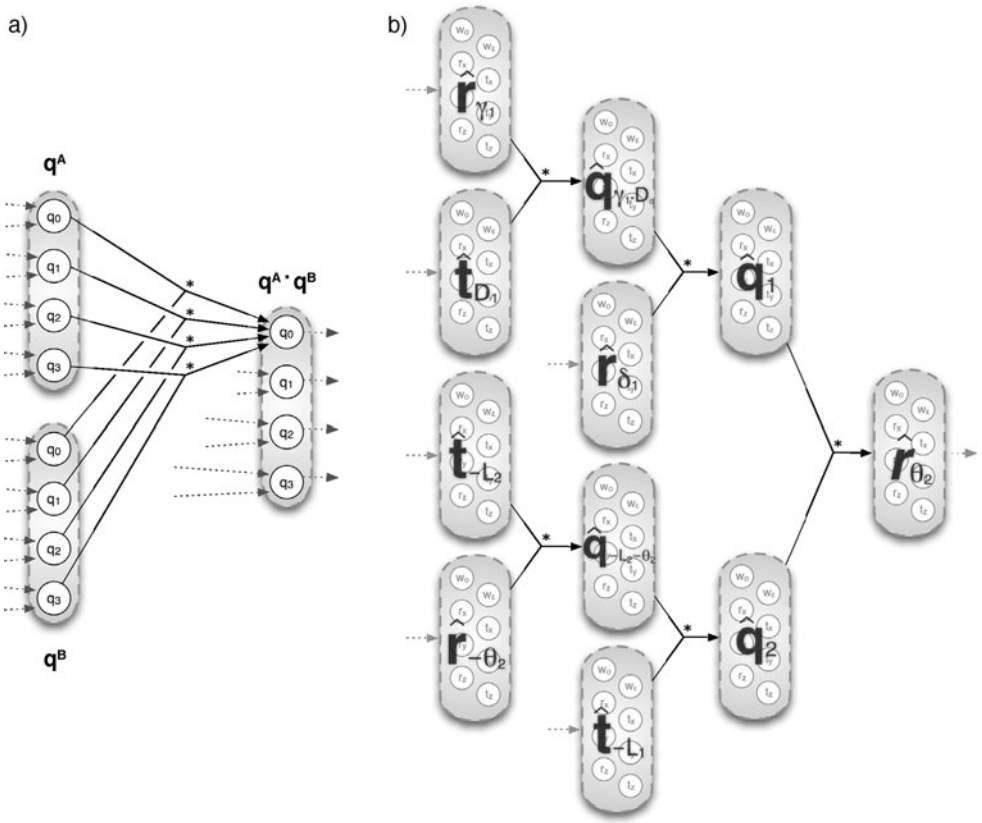

Fig. 8 Realisation of quaternion multiplication in a neural network: (**a**) Two quaternions (*left*) are multiplied. Each quaternion ($\mathbf{q}^A$, $\mathbf{q}^B$, $\mathbf{q}^A * \mathbf{q}^B$) is represented by four units. Multiplication of the values (indicated by the *asterisk*) and the following linear summation in the units at the right provides quaternion $\mathbf{q}^A * \mathbf{q}^B$ as a result. (**b**) Example of the connections describing the equation $\hat{\mathbf{r}}_{\theta_1} = \hat{\mathbf{r}}_{\gamma_1} \hat{\mathbf{t}}_{d_1} \hat{\mathbf{r}}_{\delta_1} \hat{\mathbf{t}}_{l_2}^{-1} \hat{\mathbf{r}}_{\theta_2}^{-1} \hat{\mathbf{t}}_{l_1}^{-1}$ using dual quaternions

nections of the kinematic inner network off ($d = 0$), the model works like a dynamic forward model. If a position change is given as an input to the network, depending on the current state of the system (i.e. the current position and velocities of the segments), the network predicts a new arm position. The velocity damping factor describes a property similar to the inertia of the arm segments introducing low-pass properties.

– When both damping factors are greater then zero, both parts of the network including all recurrent connections interact. The network can be used to solve inverse problems and at the same time allows access to the variables related to dynamic properties.

2.3.1 Realisation in neural networks

Traditional MMC networks can be easily represented in neural networks (Cruse and Steinkühler 1993) because the equations used to represent the geometric structure are based on summation and the resulting equations can be directly transferred into a weight matrix for a neural network. In contrast, for the structure proposed here, the equations describing geometrical relations use quaternion multiplication for the concatenation of transformations. How can such an operation be implemented in a neural network? Usually, neurons in neural network approaches are simplified as computational units that compute the weighted sum of their input. Different solutions have been proposed to implement the multiplication of two variables, for example, multi-layer perceptrons or architectures using spatial coding (Hartmann and Wehner 1995). The sigma-pi neuron proposed by Rumelhart and McClelland (1986) allows for an easier realisation. Sigma-pi units are somewhat more complex units than summation units. A sigma-pi unit is composed of two processes: the central part is—again—the summation of the weighted inputs. But prior to entering the unit as such, the incoming connections can be multiplicative. Thus, a sigma-pi unit realises the summation of products. Sigma-pi units may represent biological mechanisms like axo-dendritic synapses or pre-synaptic inhibition. Figure 8(a) shows how such units can be used to realise quaternion multiplication. Figure 8(a), for simplicity, shows the case for quaternions, whereas Fig. 8(b), as explained below, depicts, in a more abstract way, the realisation of a complete equation using dual quaternions. In Fig. 8(a) each quaternion is represented by four units, one for each value of the quaternion. Following the multiplication table (see Appendix, (8)) for quaternions, the multiplication can be directly implemented as shown explicitly for the first values (the computation of the following values can be obtained in the same way, but is not shown for better readability). This realisation of the multiplication of quaternions can now be used to implement the equations of the MMC approach.

In dual quaternions, 2 times 4 units are required. Multiplication can be realised in the same way as for quaternions. As an example, the network for (6) ($\hat{\mathbf{r}}_{\theta_1} = \hat{\mathbf{r}}_{\gamma_1}\hat{\mathbf{t}}_{d_1}\hat{\mathbf{r}}_{\delta_1}\hat{\mathbf{t}}_{l_2}^{-1}\,\hat{\mathbf{r}}_{\theta_2}^{-1}\,\hat{\mathbf{t}}_{l_1}^{-1}$) is shown in Fig. 8(b). Every variable transformation is explained by five or six other transformations represented through dual quaternions. As a consequence, four or, as in our example, five multiplications are needed. The first intermediate result $\hat{\mathbf{q}}_{\gamma_1,D_1}$ describes the first diagonal, the next one below describes the second segment $\hat{\mathbf{q}}_{\theta_2^{-1},L_2^{-1}}$. The results of the multiplication appearing on the next level (counted from left to right) are also used by other equations.

The computation of the equations necessary for the complete network can be simplified. First, most of the computations can be performed in parallel. Second, while each multiplication requires a dual quaternion for storing the result, these intermediate results are shared by different equations. Therefore, not every equation for each variable needs additional units, because several dual quaternions always occur in the context of one specific other dual quaternion. This is the case, for example, for the two dual quaternions that describe the orientation and the dimension of one side of one of the triangles that were used to set up the equations. Third, inversion of a dual quaternion (see (17) in the Appendix) corresponds to a change of the sign and, in a neural network, could be realised by inhibitory connections.

3 Three-dimensional arm example using dual quaternions

In this section, results are presented that simulate 3D movements of a three-segmented arm produced by a MMC network using dual quaternion representation. Two general questions are addressed. First, the convergence of the network is analysed. It is difficult to proof the convergence for the dual quaternion MMC network because non-linear functions are involved. Therefore, the behaviour of the network for a large set of movements is analysed. The simple kinematic MMC network is used to produce movements covering the whole workspace showing the convergence properties of the network. This first set of simulations also illustrates the major shortcoming of the kinematic MMC networks concerning the relaxation of the network. The network starts with high velocities and then slows exponentially down. Therefore, the second set of simulations addresses how the dynamic extension can overcome this problem. In these experiments, the performance of the network for a human arm reaching task is evaluated and it is shown that the network complies with characteristic criteria for human arm movement.

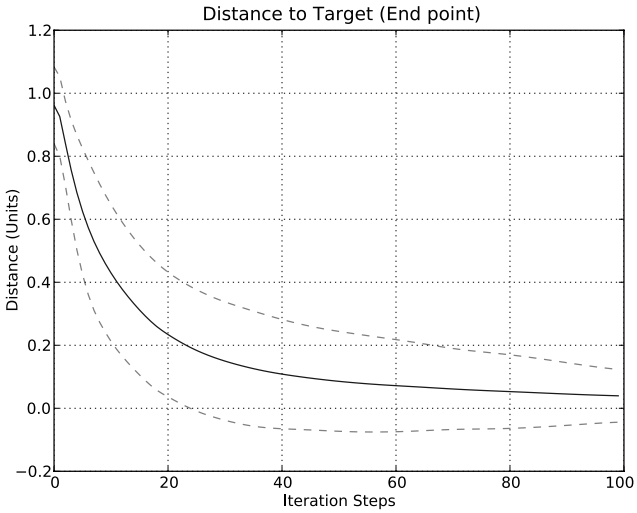

Fig. 9 Normalised distance between the tip of the arm and the target position over time. For all 1260 movements the mean normalised distance is calculated for each iteration step. The distances are normalised with respect to the distance between starting position and target position (The *dashed lines* show the standard deviation around the mean value. The standard deviation is quite high in the beginning—some movements consist of an folding and unfolding of the arm taking more time for relaxation)

3.1 Convergence of the dual quaternion MMC network

In the first experiment, the inverse kinematic problem has to be solved for a manipulator consisting of three segments of equal length (1 unit) which are connected by ball joints. This results in 9 degrees of freedom. In this positioning task the arm therefore is redundant and has supernumerous degrees of freedom. The network was constructed as explained above (the damping factor was $d = 10$ for the joint variables and $d = 2.5$ for all the other variables. The velocity control is turned off, $d_{vel} = 0$). The goal of the simulation is to show that the network is able to solve the inverse kinematic problem and to come up with a geometric correct solution independent of the starting and end point. Every simulation consists of a movement with the tip of the manipulator from a the starting point to a target point. As points, we used three sets of twelve different points which are equally distributed around the base of the manipulator. The first set consists of twelve points which all lie on a sphere with a radius of three units which equals the overall length of the arm. That means, this points are reached by the arm when the arm is fully stretched and oriented towards this points. The distance between neighbouring points is identical—the points are the corner points of a icosahedron. The same holds true for the second and third set of points used in the simulation, with the only difference being that the distances from the base of the manipulator to the points are changed. For the second set of twelve points the distance to the base is two segments length and for the third set of points it is one segment length.

Simulations were performed in which each of the 36 points was used as a starting point and from there movements to the remaining 35 points where performed.

Before each run the arm was moved from the initial configuration (fully stretched) over 100 iteration steps to the starting position. The arm always reached the starting position during that time. Then the movement started towards the target position and was measured for 100 simulation steps. Overall 1260 simulation runs (35 simulations from 36 different starting points) have been executed. In principle, many of the start-target combinations are rotationally invariant. But as non-linear functions are involved and the starting posture can differ between these start-target combinations—at least for the starting postures in which the arm is not fully stretched—it can not in general be assumed that these combinations are also invariant in the dual quaternion representation.

In general, the network solves the inverse kinematic problem for all combinations of starting and end points covering movements through the whole workspace of the manipulator. The network has no problem to execute all the movements between the targets, even for diagonal movements for which the manipulator has to cross its base position in a very short distance, meaning the arm is strongly folded. All movements show the same characteristics. The target point is approached initially very fast. Over time the distance between end effector and target is exponentially decreasing (the manipulator overshoots the target usually a little bit and in plots from an individual movement one can see this as a small damped oscilation). In Fig. 9, the mean normalised distance between the tip of the manipulator and the target point is drawn, showing this characteristic. The path of the end-effector follows a slightly curved line (individual movements are not shown for the first experiment, but the same properties hold true for the movements shown below in the second set of experiment on more biological movements). While it is not the shortest path in Cartesian Space, it is an approximation to the path which minimises joint movements. The shape of human arm movements is qualitatively similar. Human arm movements have been assumed to follow straight lines, but more recently it is recognised that movements can deviate from straight lines. Human arm movements seem to follow straight lines only when being restricted to two dimensions or when small movements are made. In contrast, reaching movements spanning three dimensions and larger parts of the working space show more curved trajectories (Atkeson and Hollerbach 1985).

Looking at the velocity profile of the tip of the manipulator shows—similar as for classical Cartesian MMC networks—an initial high velocity which in principle is then decreasing exponentially (see Fig. 10). These movement characteristics are not in agreement with that of biological

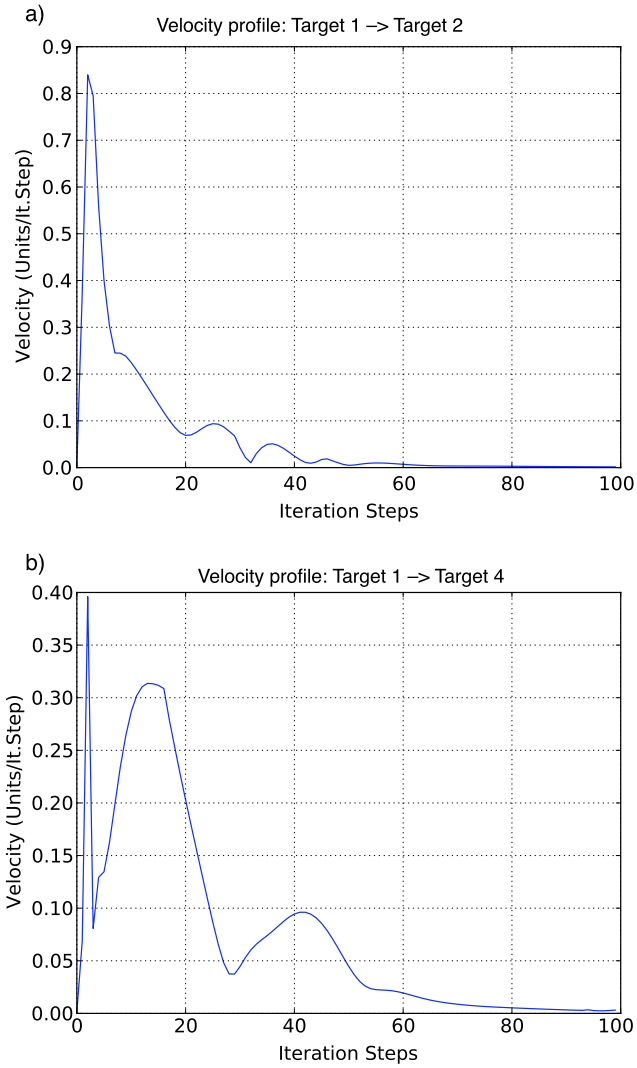

Fig. 10 Velocity profiles for two movements. In both cases the start and target point are lying on the border of the working range. The arm is therefore fully stretched to reach this position. In (**a**) the angle between the arm pointing to the starting position and the arm after having reached the target position is ninety degrees. Such a small movement shows the typical exponential decreasing velocity. In (**b**) the angle between the arm pointing to the starting position and the arm after having reached the target position is 180 degrees. In this movement the arm crosses the whole workspace. Movements covering large parts of the workspace show velocity profiles with multiple peaks as during such movements the arm performs several submovements. At first the arm is quickly folded, then during a second phase mainly reoriented (in this phase the tip is moving slowly as it is very near to the base), and at last the manipulator is unfolding towards the target with an initial high velocity again

movements. In the next section, a solution for more biologically plausible movements—given through bell-shaped velocity profiles—as has been applied in classical MMC networks (Schilling 2009) and has been explained above will be applied to the dual quaternion network in a special movement task. To conclude the first set of simulations: Until

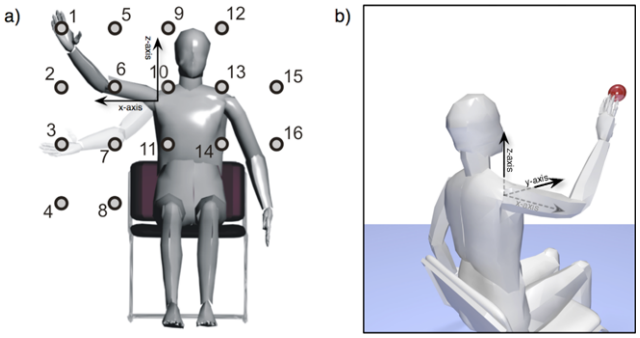

Fig. 11 Set-up of the experiment. The target positions are taken from Bockemühl et al. (2010). The task is to reach for different targets, beginning at a predefined starting position. *Left* (**a**): different target positions, front view. *Right* (**b**): view from behind and to the right of the reaching subject, as used in the following figures (target 6 is shown)

now, we have shown that dual quaternion MMC networks converge for points covering the whole workspace.

3.2 Control of a human-arm like configuration including velocity equations

In the second set of experiments the extension of the MMC network through equations including the joint velocities is analysed. In this case, the network represents an arm comparable to a human arm (upper arm 0.3 m, forearm 0.3 m and hand 0.15 m) with however 9 degrees of freedom, i.e., with the elbow as well as the other joints in the shoulder and in the hand having three degrees of freedom instead of only one or two, as is the case for a elbow or the wrist joint in a real human arm. (Simulation of a seven DoF arm was also performed but required more parameters to be adjusted. In this case, for a joint with only one degree of freedom the resulting dual quaternion is afterwards projected onto the axis of rotation of the joint. Using appropriate damping factors, the results were qualitatively the same.) The task was to generate movements towards a set of prespecified goals. In the starting position, the upper arm hangs vertically down from the shoulder, about parallel to the upper part of the body of the subject sitting on a chair. The forearm and the hand are oriented horizontally, pointing towards the front. The starting position and the 16 target points are taken from an experimental set-up investigating human arm movements (Bockemühl et al. 2010). Target positions are situated in a plane parallel to the x–z plane (see Fig. 11), z describing the vertical axis and x describing the transversal axis. Therefore, targets require movements to the front, left and right, as well as up and down.

Two different types of MMC networks were used and compared. At first, a MMC network was set up in which the geometric relations were described by dual quaternions as explained in the previous section (the damping factor was $d = 10$ and $d_{vel} = 0$). In the second approach, this MMC network is extended by additional equations containing dynamic relations of the manipulator, in our case referring to the velocity and acceleration (which can be introduced in the same way as velocities, see Schilling 2009) of joint movement. The simple extension introduces a low pass property, meaning that the velocity changes smoothly over time. Of course, other characteristics could be used and additional equations could be included in the velocity calculation, e.g., to approximate a muscle like behaviour. But, as will be shown in the results, the simple extension already circumvents the problem of the exponential decreasing velocities and produces more biological plausible velocity profiles (see Schilling 2009, for details). The damping factors for these equations will be correspondingly termed velocity damping factors d_{vel} and acceleration damping factors d_{acc} (damping factor $d = 10$, for the velocity, damping factor $d_{vel} = 2.5$, and, for the acceleration, damping factor $d_{acc} = 1$). The general characteristics of the MMC network are not changed by this extension. The dynamic equations are affecting the temporal progress of the network. The reached end position and the used trajectory remains basically unaffected. (See for a comparison of both approaches below and in the electronic supplementary material. In addition, Schilling (2009) compares the behaviour of the network for different velocity damping factors and is mainly finding changes in the velocity profiles, while trajectories are only slightly affected.)

Both MMC networks were able to solve all inverse kinematic problems, producing smooth trajectories within 30 iteration steps in all 16 cases. Here the comparison between both MMC versions will be shown for only one target, target 6, because there was no qualitative difference to be observed for the other 15 targets (the corresponding results can be found in the supplemental material, Sect. 1—for both types of network all the movements and velocity profiles are shown and compared). Figure 12 shows the movement of the arm controlled by the kinematic network and Fig. 13 shows the same for the network including dynamic equations. The main difference between the two versions can better be seen when regarding the velocity profiles (see Fig. 14). For the kinematic MMC approach the velocity is high in the beginning and then decreases exponentially, while in the dynamic case there is a more bell-shaped velocity profile with a broader distribution of high velocities. (Again, see the velocity profiles in the electronic supplementary material, Sect. 2, for results on the other targets.) Furthermore, the end effector overshoots the target position. Such an overshoot could of course be compensated by applying a control law being somewhat more sophisticated than the simple low-pass characteristic used here which results from the application of constant damping values.

Nonetheless, comparing the distances to the target position, both approaches end up in the vicinity of 1 cm of the

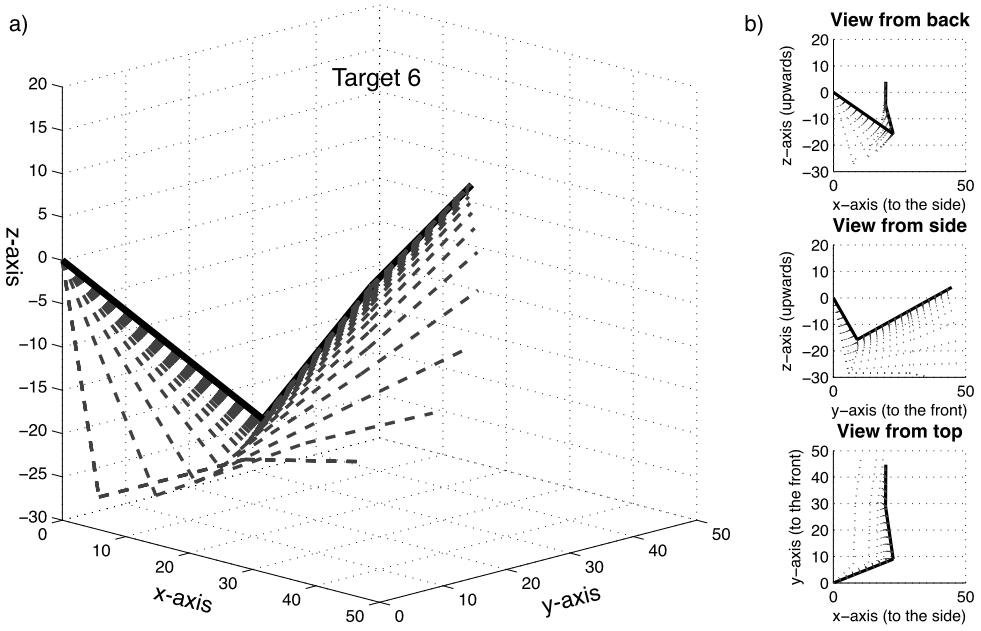

Fig. 12 Arm movement controlled by the kinematic MMC network representing joint configurations by dual quaternions. (**a**) arm position shown for every second iteration step (*dashed lines*, end position is plotted by *solid lines*), view as in Fig. 11(b). (**b**) Same data seen from back, from side and from top. For coordinates see Fig. 11(b)

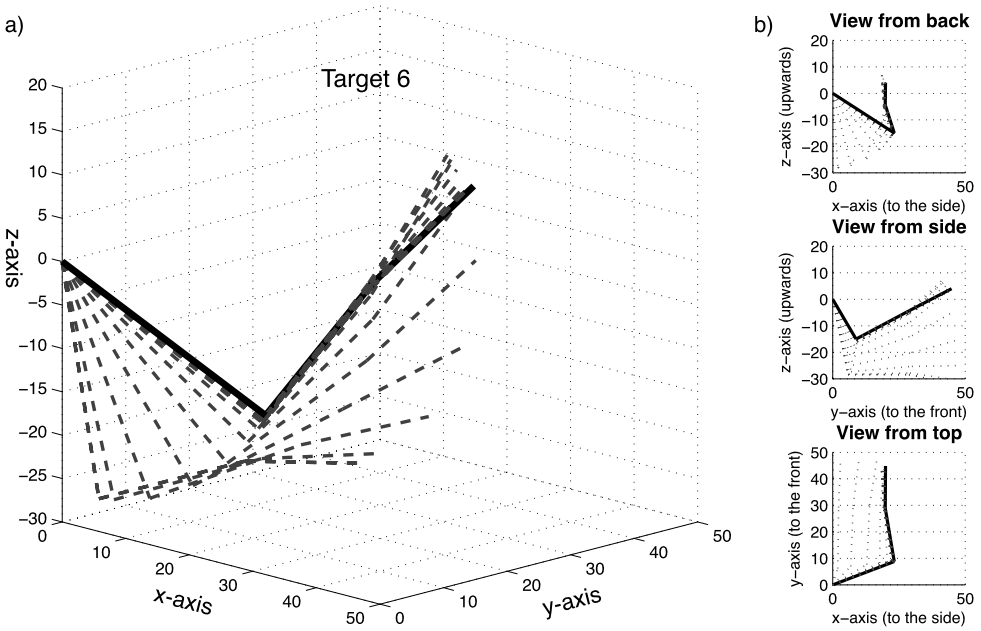

Fig. 13 Arm movement controlled by the dynamic MMC network representing joint configurations by dual quaternions. (**a**) arm position shown for every second iteration step (*dashed lines*, end position is plotted by *solid lines*), view as in Fig. 11(b). (**b**) Same data seen from back, from side and from top. For coordinates see Fig. 11(b)

target after around 25 iteration steps. The dynamic approach, however, passes the target the first time after 15 iteration steps.

For one more target the movements resulting from the two different network versions are shown. Figure 15 shows the behaviour of the kinematic MMC network when approaching target 4 and Fig. 16 shows the movement when using the dynamic version reaching for the same target. Results for all the targets and comparisons of the velocity profiles are given in the supplemental material.

4 Discussion

The study of reactive systems in the strict sense (Mataric 1999, 2002) concentrated on systems that were assumed to have no internal states and focussed on embodied systems acting in an environment. In contrast, the understanding of mechanisms underlying cognition is assumed to take internal representations into account. Considering internal representations, it appears to be plausible that their basis is formed by a model of the own body, which may later be

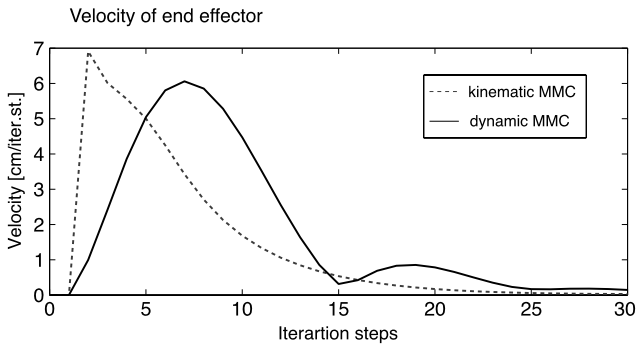

Fig. 14 Velocity profile of the end effector. Comparison of movement characteristics of the kinematic MMC and the MMC network including equations for representing velocities

extended to include properties of the environment (Cruse 1999, 2003; Schilling and Cruse 2008). To learn how internal representations might be realised and applied by a neuronal system, concentration on motor control tasks appear to be a good starting point, because the domain of motor control is considered a central paradigm for the application of internal representations (e.g., Frith et al. 2000). For simple examples of behavioural elements, one might think of a swing movement or a stance movement performed by a leg during walking or a reaching movement performed by an arm. How might neuronal systems be designed that are able to control such behavioural elements? Most simple solutions are given by fixed action patterns, movements elicited by a stimulus and not changeable by further sensory feedback after the movement has once started. Neural networks controlling such fixed action patterns may be regarded as containing implicit representations of the body, for example. More flexible solutions are given by the application of feedback controllers. The feedback might be applied using a positive (Schmitz et al. 2008) or a negative sign, and it might concern various entities such as position (and derivatives), force or mixed information. Feedback controllers might be equipped with inverse models and forward models, in particular when the system to be controlled contains redundant degrees of freedom (Kawato and Gomi 1992). A most intensively studied paradigm concerns reaching movements. Given a starting configuration of the arm, the hand has to be moved to a (usually visually) given point in space. The many hypotheses being proposed as solutions for the task to find a trajectory for the hand and an end position of the arm can be classified into three groups. Two of them have been most intensively studied. First, the equilibrium point controllers and secondly, (different types of) minimisation procedures (minimum torque (Uno et al. 1989), minimum jerk (Flash and Hogan 1985), minimum work (Soechting et al. 1995) and geodesics (Biess et al. 2007)). Both approaches require inverse models. Minimisation approaches require the computation of the complete movement before execution is possible which is not necessary for the equilibrium point controllers.

Models of the third group that are characterised by the "passive motion paradigm" (Mussa-Ivaldi et al. 1988) also do not rely on the inverse kinematic computation in advance. Application of the passive motion paradigm requires a model that (implicitly or explicitly) represents the kinematic properties of the body to be controlled. Following this paradigm, the tip of the hand of this "internal model" can then be moved to any desired endpoint and—as in the case of the arm of a puppet—the arm segments follow passively, providing an internally simulated solution to the problem. The movements of the simulated joints can then be used to directly control the movements of the corresponding actual joints.

We are aware of two approaches following this paradigm, the knowledge model of Rosenbaum et al. (1993, 1995, 2001) and the MMC network studied here. Both approaches share the advantages of not requiring the precomputation of the complete movement, being able to deal with extra degrees of freedom and finding solutions even if some extra degrees of freedom may be controlled externally (e.g., by fixing these joints). Compared with the knowledge model, to date the MMC approach has the disadvantage that learning is not possible yet (apart from a solution concerning linear MMC nets in Cruse and Hübner 2008). On the other hand, the number of neuronal units required is much smaller in the MMC approach. Furthermore, as all relevant variables (joint angles, joint and endpoint positions, as well as velocities) are explicitly addressable (which also includes the possible introduction of various constraints, e.g., joint limits), the MMC model is capable of representing and exploiting all geometrically possible configurations. This property has the consequence that the MMC net can also be applied as a forward model just by correspondingly selecting the input variables.

We have addressed how a MMC model, here realised by a network based on dual quaternions, can be used for solving the inverse kinematic problem which is in robotics usually thought of as the hard problem. Closed-form solutions to the inverse kinematic problem are based on inverting the Jacobian matrix. While this can only be derived for simple manipulator structures or specific industrial robots underlying certain constraints (like positioning of axes), closed-form solutions only exist for a small set of manipulators. This can be extended through using a pseudo-inverse (e.g., Whitney 1969). The inverse kinematic problem is also approached through numerical methods. On the one hand, there are approaches also relying on inverting the Jacobian matrices. These approaches cannot find a solution when the matrix is singular. On the other hand, there are optimisation approaches which essentially are performing a sort of gradient descent. One such example is the cyclic coordinate descent (CCD) algorithm (Luenberger 1984). CCD

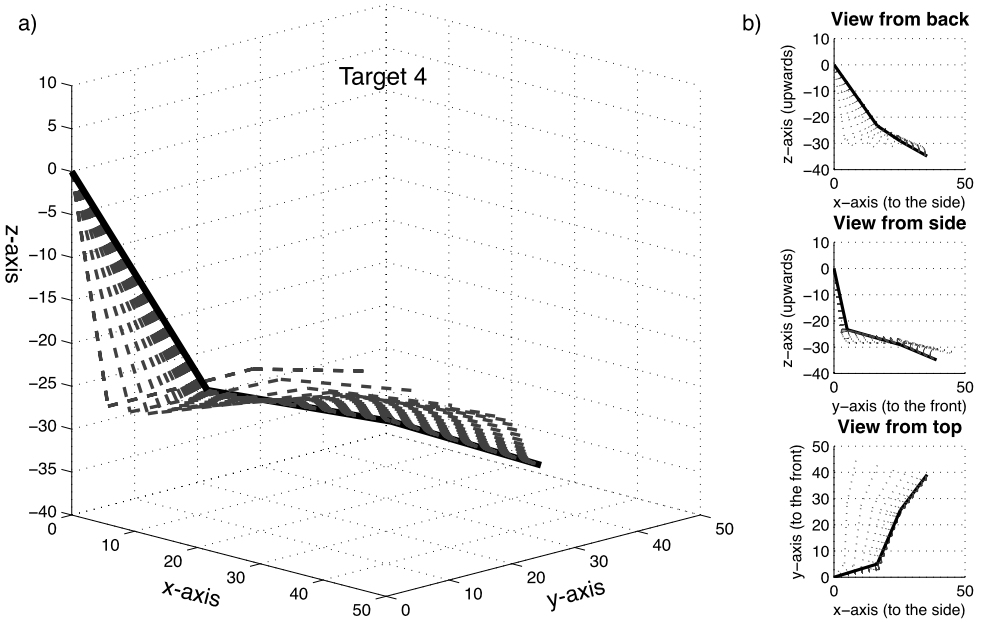

Fig. 15 Arm movement controlled by the kinematic MMC network representing joint configurations by dual quaternions for Target 4. (**a**) arm position shown for every second iteration step (*dashed lines*, end position is plotted by *solid lines*), view as in Fig. 11(b). (**b**) Same data seen from back, from side and from top. For coordinates see Fig. 11(b)

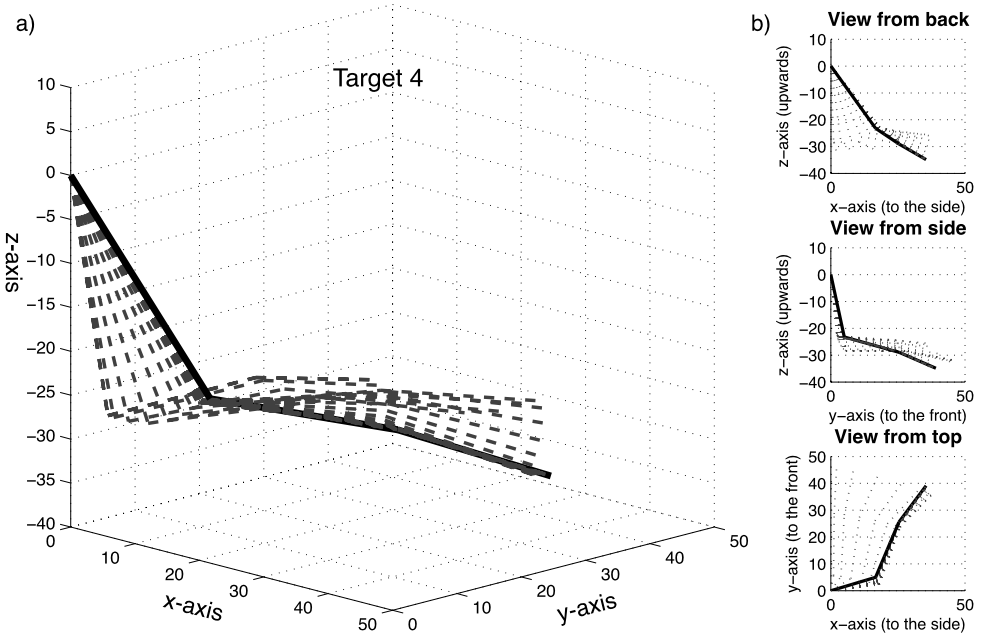

Fig. 16 Arm movement controlled by the dynamic MMC network representing joint configurations by dual quaternions for Target 4. (**a**) arm position shown for every second iteration step (*dashed lines*, end position is plotted by *solid lines*), view as in Fig. 11(b). (**b**) Same data seen from back, from side and from top. For coordinates see Fig. 11(b)

is an iterative heuristic search procedure over the joint angles. CCD individually tries to change the joint angle in order to minimise the overall positioning error. During one iteration step, it computes the position error and then steps backwards through the whole structure—starting at the most outer joint—and adjusts each joint value in order to minimise the position error. CCD has shown to rapidly find a good approximation and is therefore often combined with other methods which then take over and find an exact solution (Wang and Chen 1991). The complexity of the approach increases dramatically with the complexity of the structure.

While the MMC approach is comparable in that for each joint individually an error term is reduced, the MMC approach is relying on local relations and not one global error measurement. The main difference is that the MMC network is distributing the error onto different variables: In the inverse kinematic the end effector position is changed. The joints try to compensate for the error and at the same time the error is distributed to the additional variables (diagonals) which contribute in the next iterations in guiding the overall optimisation process. This has the disadvantage that in the MMC approach the number of used equations increases

in more complex structure. But this can be counteracted through distributing the complexity of the manipulator onto different levels of abstraction (Kindermann and Cruse 2002; Schilling and Cruse 2007). The advantage of the MMC approach is that each of the multiple computations contributing to the calculation of a variable can be easily derived and calculated. Especially all these computations are independent and can be carried out at the same time. Unfortunately, this makes it difficult to compare efficiency of MMC networks and the CCD approach, as procedures like CCD are working in strict sequence while the MMC network is a neural network allowing for massive parallelisation. The second major advantage of the MMC network is given by its autoassociator capacities which allow this network to be applied in addition to forward or any mixed kinematic problem in a flexible manner.

As mentioned in the Introduction, Morasso and Sanguineti (1994) have introduced such a "holistic" model by connecting an explicit inverse model with an explicit forward model. Wolpert and Kawato (1998) proposed a different approach. In their "MOSAIC" model for each individual behavioural element, for example, a leg performing a swing, a hand lifting a heavy object, a hand lifting a light object etc., a pair of models is assumed to be implemented, each consisting of an inverse model and a forward model necessary for the corresponding task. An advantageous property of applying such dedicated models is that during performance of the action, the inverse models of several behaviours can be active in parallel and the error (i.e., the difference between prediction and sensory feedback) can be used to decide which is the model that best matches the task at hand. A disadvantage compared with the MMC model is that only already stored behaviours (or a weighted combination of stored behaviours) can be performed and estimated. Completely new behaviours that may result from unusual exploitation of the extra degrees of freedom may not be able to be produced (for motor control or for planning) or be evaluated in a sensible way. A related problem is that linear averaging may lead to erroneous results.

There is another critical difference between the two approaches. When the number of behavioural elements being stored increases, in the MOSAIC model the number of inverse/forward model pairs increases correspondingly. As most behaviours require movement of the complete body (even if only an arm appears to be moving), a large number of complete body models may be required. Therefore, as an alternative approach we propose to use a unique body model. The individual behavioural elements may then be driven by separate networks that, however, share this model. Support for a separation of the body model from task-specific models has been provided by Cothros et al. (2006). In the experiment people were trained to make targeted movements with their arms while holding a robotic device through which a novel force field was applied to the hand. While adaptation to a given force field has been observed many times before (Shadmehr and Mussa-Ivaldi 1994) and has been termed motor learning (Kawato 1999), Cothros et al. (2006) studied adaptations to different force field and were especially interested how these adaptations influence each other. The result was that subjects who returned to a known force field were able to completely retain their previous learning. This result supports the idea that during learning not an existing model of the complete body plus the current task is adapted—or even newly learned—but only an additional model of the object or the associated movement dynamics is constructed. Additional support comes from Davidson and Wolpert (2004) and more and more evidence is emerging, favouring a modular organisation of internal models (Imamizu and Kawato 2008; Ghahramani and Wolpert 1997; Krakauer et al. 1999).

As for its evolutionary plausibility, this unique body model might well have been developed through the merging of several of such dedicated networks, because application of a unique model allows to minimise the number of neurons necessary and to avoid errors resulting from inadequate averaging between different modules. Actually, many results indicate that internal models are widely distributed all over the brain. For specific behaviours, the cerebellum is proposed to implement forward and inverse models (Wolpert et al. 1998). For coordination of hand and eye, Makin et al. (2008) propose in addition the posterior parietal cortex, including the intraparietal sulcus and the premotor cortex. Furthermore, some neurological phenomena indicate that the coupling between sections of the body model can be weakened (e.g., OBE (Blanke et al. 2004, 2005), see Schilling and Cruse 2008 for discussion).

To summarise: we propose not only to apply specific and task-oriented models that can serve specific actions, as found in reactive systems, but to go one step further and think of ways models can be interconnected and related in a flexible way. To this end, we propose a universally manipulable model that is able to represent both active and passive movements that the real body is able to perform. The model can be used for motor control (as an inverse model as shown here), for perception and as a simulator for planning ahead, i.e., applying actions to the model being decoupled from the body (Hesslow 2002). These actions can be controlled by specific modules representing knowledge required earlier (i.e., by procedural memory) or by newly invented behavioural elements.

The next step is to realise a network that is able to represent a complete body, for example, the body of a hexapod walker with 18 joints and to use this body model for planning ahead (for a review on the use of body models in robotics see Hoffmann et al. 2010). It has been shown (Kindermann and Cruse 2002; Schilling and Cruse 2007) that

this task can be solved by modularising the structure using modules (e.g., representing a leg) connected in a hierarchical manner. In this article, we only concentrated on one such module representing an arm or a leg. Based on these results, we are currently applying this approach to control a six-legged robot equipped with the faculty to use its body model for planning ahead and for finding solutions to new problems.

Acknowledgements This work has been supported by the Wissenschaftskolleg zu Berlin, the Center of Excellence Cognitive Interaction Technology, the Deutsche Forschungsgemeinschaft (DFG, grant Cr 58/11-1) and a DAAD grant to Malte Schilling. We would also express our thanks to Mitch Cohen, Berlin, for proof reading the manuscript and two anonymous reviewers for their helpful comments on an earlier version of this paper.

Appendix A: Dual quaternions

The Appendix shall, in brief, provide the necessary mathematical background on dual quaternions. As different authors use different notations, those used here and the basic calculations will be established. Quaternions, dual numbers and dual quaternions will be introduced. For further reading see, e.g., Hanson (2005) on quaternions. On dual quaternions, Kavan et al. (2008) is recommended as an introduction, whereas McCarthy (1990) provides more details.

A.1 Quaternions

Quaternion algebra was formulated by W.R. Hamilton (Hamilton 1844). This algebra represents a special case of the geometric algebras (also termed Clifford algebras).
A quaternion $\mathbf{q}$ is a 4-tupel $\mathbf{q} = w + xi + yj + zk$, with w, x, y, z being real numbers and i, j, k being the quaternion units. w is the scalar or real part of the quaternion and (x, y, z) is usually termed the vector part. Summation and subtraction are executed component wise. The quaternion units are given as $i^2 = j^2 = k^2 = ijk = -1$ and following from this $ij = -ji = k, jk = -kj = i, ki = -ik = j$. The multiplication of two quaternions can be derived from the multiplication of the units. For every two quaternions $\mathbf{q}_0 = w_0 + x_0 i + y_0 j + z_0 k$ and $\mathbf{q}_1 = w_1 + x_1 i + y_1 j + z_1 k$ the product of the two quaternions is

$$\begin{aligned}\mathbf{q}_0\mathbf{q}_1 = {} & (w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1) \\ & + i(x_0 w_1 + w_0 x_1 + y_0 z_1 - z_0 y_1) \\ & + j(y_0 w_1 + w_0 y_1 + z_0 x_1 - x_0 z_1) \\ & + k(z_0 w_1 + w_0 z_1 + x_0 y_1 - y_0 x_1) \end{aligned} \tag{8}$$

or using a vector notation this can be subsumed to $\mathbf{q}_0\mathbf{q}_1 = (w_0 + \mathbf{v}_0)(w_1 + \mathbf{v}_1) = (w_0 w_1 - \langle \mathbf{v}_0, \mathbf{v}_1 \rangle) + w_0 \mathbf{v}_1 + w_1 \mathbf{v}_0 + \mathbf{v}_0 \times \mathbf{v}_1$. The multiplication is associative and distributive, but in general not commutative. The conjugate of a quaternion $\mathbf{q}$ is defined as

$$\mathbf{q}^* = w - xi - yj - zk \tag{9}$$

For the conjugate of a product of two quaternions $\mathbf{p}$, $\mathbf{q}$ the following holds true: $(\mathbf{pq})^* = \mathbf{q}^*\mathbf{p}^*$.
The norm of a quaternion $\mathbf{q}$ is defined as

$$\|\mathbf{q}\| = \sqrt{w^2 + x^2 + y^2 + z^2} = \sqrt{\mathbf{q}^*\mathbf{q}} = \sqrt{\mathbf{q}\mathbf{q}^*} \tag{10}$$

The inverse of a quaternion $\mathbf{q}$ is only defined for $\mathbf{q} \neq 0$:

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \tag{11}$$

Quaternions can be used to represent rotations that, on the one hand, are more compact, more efficient and unambiguous compared to matrix notations, and, on the other hand, are singularity free in contrast to representations like Euler angles. A 3D rotation can be described by a rotation axis given by a vector (a_x, a_y, a_z) of unit length and by a rotation angle α. This rotation can be written as the quaternion

$$\mathbf{q} = \cos\left(\frac{\alpha}{2}\right) + (a_x i + a_y j + a_z k)\sin\left(\frac{\alpha}{2}\right) \tag{12}$$

As can be easily verified, this quaternion is of unit length. Quaternions of the norm 1 are the unit quaternions, forming a sub-group over the quaternions and forming the surface of a hypersphere S^3. The rotation group for three dimensions is the special orthogonal group $SO(3)$. There exists a mapping between the two groups, i.e., the group of unit quaternions is a double cover of the group of rotations in three-dimensional space. In other words, every rotation corresponds to two unit quaternions ($\mathbf{q}$ and $-\mathbf{q}$).

A vector (v_x, v_y, v_z) can be represented as a quaternion $\mathbf{v} = v_x i + v_y j + v_z k$, i.e., the scalar part is zero. Applying a rotation onto this vector would then be expressed as

$$\mathbf{q}\mathbf{v}\mathbf{q}^* \tag{13}$$

The concatenation of subsequent rotation operations—$\mathbf{q}$ followed by $\mathbf{p}$—is given as the multiplication of the corresponding quaternions. Analogous to the conventions used for matrix multiplications, the order of the multiplicands is determined by their frame of reference:

1. If the involved rotations are described with respect to a fixed coordinate system—e.g., the world coordinate system—then the concatenation is realised as a left multiplication, i.e., $\mathbf{pq}$.
2. In contrast to this, if the second rotation is defined with respect to the coordinate system generated by the first rotation, then the concatenation is realised as a right multiplication, i.e., $\mathbf{qp}$. This is the case for the transformations

describing the kinematics of a manipulator. For example, the rotation of the second joint is not given with respect to the coordinate frame of the manipulator base or any other fixed world coordinate system, but is given relatively to the coordinate system of the prior segment which as such is defined by the preceding rotation of the first joint.

A.2 Dual quaternions

The algebra of duals is similar to that of complex numbers. A dual number is a two tupel $\hat{a} = a_0 + \varepsilon a_\varepsilon$. a_0 is the non-dual part and a_ε the dual part with ε being the dual unit for which holds true $\varepsilon^2 = 0$. Summation and subtraction are done component wise. Multiplication of dual numbers $\hat{a}$, $\hat{b}$ produces $\hat{a}\hat{b} = (a_0 + \varepsilon a_\varepsilon)(b_0 + \varepsilon b_\varepsilon) = a_0 b_0 + \varepsilon(a_0 b_\varepsilon + a_\varepsilon b_0)$.
The conjugate of a dual number $\hat{a}$ is $\overline{\hat{a}} = a_0 - \varepsilon a_\varepsilon$.
The inverse of a dual number exists only if $a_0 \neq 0$ and equals $\hat{a}^{-1} = \frac{1}{a_0 + \varepsilon a_\varepsilon}$.

Dual quaternions can be regarded as quaternions which elements are dual numbers instead of real numbers. In other words, dual quaternions can be regarded as a sum of two quaternions $\hat{\mathbf{q}} = \mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon$. Analogous to quaternions, dual quaternion addition and subtraction is done component wise. Dual quaternions are associative and distributive, but in general not commutative. The multiplication of two dual quaternions—$\hat{\mathbf{q}}_0 = (w_0 + x_0 i + y_0 j + z_0 k) + \varepsilon(w_0^\varepsilon + x_0^\varepsilon i + y_0^\varepsilon j + z_0^\varepsilon k)$ and $\hat{\mathbf{q}}_1 = (w_1 + x_1 i + y_1 j + z_1 k) + \varepsilon(w_1^\varepsilon + x_1^\varepsilon i + y_1^\varepsilon j + z_1^\varepsilon k)$—can be directly obtained through multiplying out the dual quaternions following the introduced rules for multiplication for quaternions and dual numbers (the dual unit commutes with the quaternion units):

$$\begin{aligned}
\hat{\mathbf{q}}_0\hat{\mathbf{q}}_1 = {} & (w_0 w_1 - x_0 x_1 - y_0 y_1 - z_0 z_1) \\
& + i(x_0 w_1 + w_0 x_1 + y_0 z_1 - z_0 y_1) \\
& + j(y_0 w_1 + w_0 y_1 + z_0 x_1 - x_0 z_1) \\
& + k(z_0 w_1 + w_0 z_1 + x_0 y_1 - y_0 x_1) \\
& + \varepsilon(w_0 w_1^\varepsilon + w_0^\varepsilon w_1 - x_0 x_1^\varepsilon - x_0^\varepsilon x_1 \\
& - y_0 y_1^\varepsilon - y_0^\varepsilon y_1 - z_0 z_1^\varepsilon - z_0^\varepsilon z_1) \\
& + \varepsilon i(w_0 x_1^\varepsilon + w_0^\varepsilon x_1 + x_0 w_1^\varepsilon + x_0^\varepsilon w_1 \\
& + y_0 z_1^\varepsilon + y_0^\varepsilon z_1 - z_0 y_1^\varepsilon - z_0^\varepsilon y_1) \\
& + \varepsilon j(w_0 y_1^\varepsilon + w_0^\varepsilon y_1 - x_0 z_1^\varepsilon - x_0^\varepsilon z_1 \\
& + y_0 w_1^\varepsilon + y_0^\varepsilon w_1 + z_0 x_1^\varepsilon + z_0^\varepsilon x_1) \\
& + \varepsilon k(w_0 z_1^\varepsilon + w_0^\varepsilon z_1 + x_0 y_1^\varepsilon + x_0^\varepsilon y_1 \\
& - y_0 x_1^\varepsilon - y_0^\varepsilon x_1 + z_0 z_1^\varepsilon + z_0^\varepsilon w_1)
\end{aligned} \tag{14}$$

Differing definitions of the conjugate of a dual quaternion can be found, as dual quaternions can be regarded as quaternions over dual numbers or as a sum of two quaternions. Both interpretations individually can be used to derive conjugation. Following conjugation of a quaternion this would yield $\hat{\mathbf{q}}^* = \mathbf{q}_0^* + \varepsilon \mathbf{q}_\varepsilon^*$ and following conjugation of a dual number we obtain $\overline{\hat{\mathbf{q}}} = \mathbf{q}_0 - \varepsilon \mathbf{q}_\varepsilon$. For the application of describing geometric relations by dual quaternions and using—analogous to the usage of quaternions—left multiplication with the dual quaternion and right multiplication with its conjugate for the calculation of the transformation (see (13)), both conjugations have to be combined:

$$\overline{\hat{\mathbf{q}}^*} = \mathbf{q}_0^* - \varepsilon \mathbf{q}_\varepsilon^* \tag{15}$$

The norm for dual quaternions is defined as

$$\begin{aligned}
\|\hat{\mathbf{q}}\| &= \sqrt{\hat{\mathbf{q}}^* \hat{\mathbf{q}}} = \sqrt{\hat{\mathbf{q}}\hat{\mathbf{q}}^*} \\
&= \|\mathbf{q}_0\| + \varepsilon \frac{\langle \mathbf{q}_0, \mathbf{q}_\varepsilon \rangle}{\|\mathbf{q}_0\|}
\end{aligned} \tag{16}$$

The inverse of a dual quaternion only exits when $\mathbf{q}_0 \neq 0$:

$$\hat{\mathbf{q}}^{-1} = \frac{\hat{\mathbf{q}}^*}{\|\hat{\mathbf{q}}\|^2} \tag{17}$$

The unit dual quaternions are those dual quaternions of unit norm, i.e., $\|\hat{\mathbf{q}}\| = 1$, which means that for every unit quaternion $\|\mathbf{q}_0\| = 1$ and $\langle \mathbf{q}_0, \mathbf{q}_\varepsilon \rangle = 0$.

The dual unit quaternions with a zero dual part ($\mathbf{q}_\varepsilon = 0$) are equal to quaternions and therefore are related to rotations in the same way as are quaternions. These dual quaternions represent rotations. Furthermore, dual quaternions can be used to represent translations. A translation about a vector (t_x, t_y, t_z) is represented as the quaternion $\hat{\mathbf{q}} = 1 + \frac{\varepsilon}{2}(t_x i + t_y j + t_z k)$ which is also a unit dual quaternion.
Rigid transformations in general are combinations of rotations and translations. Such concatenations are achieved through dual quaternion multiplication (for the correct sequence of multiplicands see above). As an example, a rotation expressed through the dual quaternion $\hat{\mathbf{q}} = \mathbf{q}_0$ should be followed by the translation $\hat{\mathbf{p}} = 1 + \frac{\varepsilon}{2}(t_x i + t_y j + t_z k)$ (defined in the root coordinate system from which follows that the translation describing dual quaternion is left multiplicated):

$$\left(1 + \frac{\varepsilon}{2}(t_x i + t_y j + t_z k)\right)\mathbf{q}_0 = \mathbf{q}_0 + \frac{\varepsilon}{2}(t_x i + t_y j + s t_z k)\mathbf{q}_0$$

In analogy to the notation of quaternions, every unit dual quaternion can be written as:

$$\hat{\mathbf{q}} = \cos\left(\frac{\hat{\theta}}{2}\right) + \hat{\mathbf{s}} \sin\left(\frac{\hat{\theta}}{2}\right) \tag{18}$$

where $\hat{\mathbf{s}} = \mathbf{s}_0 + \varepsilon \mathbf{s}_\varepsilon$ is a unit dual vector and $\hat{\theta} = \theta_0 + \varepsilon \theta_\varepsilon$. This formulation can directly be interpreted as describing a

screw motion. Following Chasle's theorem (Daniilidis 1999; Chasles 1830) every rigid transformation can be described as a screw motion, i.e.,

- a rotation of $\frac{\theta_0}{2}$ around the rotation axis given by the vector $\mathbf{s}_0$,
- a translation of $\frac{\theta_\varepsilon}{2}$ along the rotation axis $\mathbf{s}_0$,
- and defined by the position of the rotation axis in space given through $\mathbf{s}_\varepsilon$ which is the moment of the axis and which can be obtained by $\mathbf{a}_\varepsilon = p \times s_0$ with $\mathbf{p}$ being a vector to any point on the axis.

Appendix B: Computation of the mean: quaternion interpolation

One disadvantage of most representation formalisms concerns the question of how to interpolate between two states. While the translational part is trivial, the rotation is problematic. Therefore we want to concentrate at first on how to blend an arbitrary number of rotations before extending this algorithm to transformations in general. Homogenous transformation matrices are usually seen as the standard representation formalism. Nonetheless, blending between two individual matrices is usually done via a detour. Computer graphics heavily deal with such interpolations and use quaternions to interpolate. Even though this requires one additional transformation from the rotation matrix to a quaternion and another one back from the quaternion representation to a matrix, the advantages outweighs these additional computations. Furthermore, this procedure is more efficient and computationally less costly.

Interpolation of quaternions can be easily imagined: Each unit dual quaternion representing a valid rotation corresponds to a point on the four-dimensional unit hypersphere. The shortest circular arc connecting these two points should now be found. To illustrate this problem, we can think of a two dimensional unit circle in the x–y plane. If we want to find the point lying in the middle of the two, one way is to find a rotation which maps one point onto the other. For the two-dimensional case the centre of rotation is simply the origin and the angle of rotation is the enclosed angle of the two radii. The point lying in the middle can now be calculated by applying a rotation with half the angle on the first point. In the same way, other points between the two points on the unit circle can be found and can be used to interpolate between these two. This procedure can be extended to the four-dimensional hypersphere. In the case of the four-dimensional hypersphere, the center of rotation can not only be described by the origin, but must be represented by a rotation axis. Together with the origin, the two points on the unit hypersphere span a three-dimensional hyperplane. The rotation takes place in this hyperplane around the origin. The rotation axis can therefore be described by a vector which is orthogonal to the hyperplane—the normal. The rotation angle is, again, the angle enclosed by the two radii and for interpolating between the two points a rotation of the first point around the same axis by an amount of the original rotation angle can simply be applied. This transformation describing the transformation between two quaternions $\mathbf{p}$ and $\mathbf{q}$ can be computed as $\mathbf{p}^*\mathbf{q}$. The Spherical Linear Interpolation algorithm (SLERP) (Shoemake 1985) follows this procedure to blend two quaternions which each describe a rotation. This approach produces shortest path, constant speed and shortest path invariance (Shoemake 1985; Kavan et al. 2007). However, the major disadvantage of this algorithm is that it can not be extended to more than two quaternions. When we want to interpolate between more than two quaternions, we have to start with two and afterwards we can interpolate the result with the next quaternion. Unfortunately, the result of this procedure depends on the sequence and is not very efficient.

An easy solution which is at the same time fast and efficient has been proposed by Govindu (2004), Kavan and Žára (2005). Interpolation is realised component-wise over the set of quaternions. It is, therefore, named Quaternion Linear Blending (QLB). Afterwards, the result is normalised through projecting it onto the unit hypersphere. Again, this can be better illustrated for the two dimensional case: the point lying between the two other points is constructed as lying on the straight line connecting both points. Then, the point is projected to the unit circle. This method can be extended to an arbitrary number of quaternions. It uses the shortest path and is coordinate invariant. But due to the nonlinearity of the projection onto the unit sphere, it does not result in constant velocities. Kavan and Žára (2005) have computed the upper bound for the difference of an interpolation angle produced by the linear component-wise method and a SLERP like method. The upper bound is 8.15 deg, but in practice it is usually much smaller.

How can these methods for blending between quaternions be extended to dual quaternions? For both methods presented, there exists an extension to unit dual quaternions. The Screw Linear Interpolation (ScLERP) is the extension of the SLERP algorithm. A dual quaternion is not simply describing a rotation around an axis located in the origin. As mentioned above, a dual quaternion describes the parameters of a screw motion. In addition to the axis of rotation and the rotation angle, these parameters describe the relative position of the rotation axis, given by a translation vector, and a translation along the rotation axis. In the same way SLERP is interpolating two rotations by weighting the connecting transformation matrix between these two quaternions. This means SLERP blends rotation axis and rotation angle described by the two quaternions individually over time, ScLERP blends dual quaternions by interpolating individually the parameters describing the screw (for details

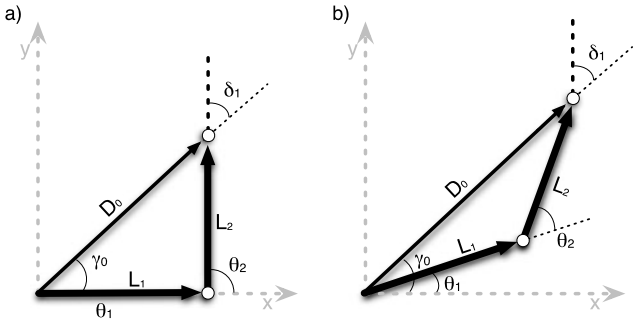

Fig. 17 Graphical representation: two segments, upper arm (L_1), lower arm (L_2), D_1

see Kavan et al. 2006, 2008). ScLERP preserves the properties of shortest path, constant velocity and coordinate invariance. Again, a simpler solution is the component-wise interpolation between a set of dual quaternions (Dual quaternion Linear Blending (DLB)), being easier to compute and being extendable to more than two dual quaternions. Kavan et al. (2006) proposed an iterative algorithm for computing a valid dual quaternion as the result of a weighted set of dual quaternions which would be the equivalent to QLB for the dual quaternion case. For our case the component-wise interpolation is sufficient, as the interpolated quaternions stemming from multiple computations will converge over time due to the iterative nature of the MMC approach.

In short, the two approaches differ as SLERP and ScLERP handle individually the parameters describing the rotation, or the screw describing the transformation, while the linear blending works component-wise. As for the linear blending of quaternions, a normalisation is also needed for DLB which can be computed easily. The upper bound for the derivation of DLB from ScLERP is for the rotational part—as for quaternions—8.15 deg and for the translational part 15.1% of the translation between the two dual quaternions involved. DLB is therefore not producing movements of constant speed, but regarding the usually occurring small differences, it is acting at almost constant speed. DLB generates the shortest path and coordinate invariance holds true (Kavan et al. 2006, 2007, 2008).

Appendix C: Compensation of displacements in rotational joints

On the one hand, the MMC network can describe a kinematic chain by using dual quaternions and solve in this way the forward kinematic problem. On the other hand, for solving the inverse kinematic problem the mean of multiple computation principle is used by computing the mean as an interpolation over a set of dual quaternions. In the introduction of the MMC vector network, we have mentioned the need for additional constraints which are applied between the iteration steps of the recurrent network. This constraints were used in order to keep segments length constant or to monitor joint angles because the recurrent network searched for an attractor by distributing the external introduced disturbance over all variables without differentiating between vectors which should keep their length and ones that are flexible (like the diagonals).

The computation of the kinematic chain through dual quaternion can directly address this problem of changing segment lengths: the multiple computations act on concatenations of dual quaternions. The segments can be easily fixed by keeping the according translation fixed. The joints on the other hand should only represent rotations and there should be no translational portion in dual quaternions representing a joint. When disturbing a quaternion MMC network—like in an inverse kinematic task introducing a new target position—one introduces an error term which is spread over all variables which are the dual quaternions representing the rotation in the joints and rotation as well as translation of the diagonals while the translations along the segments are kept fixed. As a result, it can happen that this lead to translational shares in the dual quaternions representing a joint.

To illustrate the problem: when looking at the multiplication table of dual quaternions (see (14)) what one can basically see is that combining two arbitrary unit dual quaternions results in a dual quaternion, in which the dual part depends on both—rotational shares and translational shares of the dual quaternions—, but the real part of the dual quaternion which represents the rotation is just depending on the two involved rotations and does not take the translational parts into account. This means, that the rotational part, in which we are interested when computing the joint values for solving the inverse kinematic problem only depends on the rotational parts of all influencing transformations. One can compare this to calculating the angular sum in a triangle. For the planar case this is exactly what we are doing. For any arbitrary case of three-dimensional rotations the MMC principle reduces one equation to a triangle in which then all rotations are concatenated in order to add up to an identity rotation.

To illustrate the solution in general, we are concentrating again on a geometric illustration of the problem for one equation, i.e., one triangle. As an example, we take the first two segments and the diagonal. When the network is now describing a current configuration (see Fig. 17(a)) all the rotation describing dual quaternions have no translational shares. What happens, if we now are just elongating the diagonal? The only value that is changed is the length of the translation along the diagonal. What the network should do is an extension of the arm which can be only done by moving the two joints (see Fig. 17(b)). But none of the rotations is affected by the modification and therefore, as argued above,

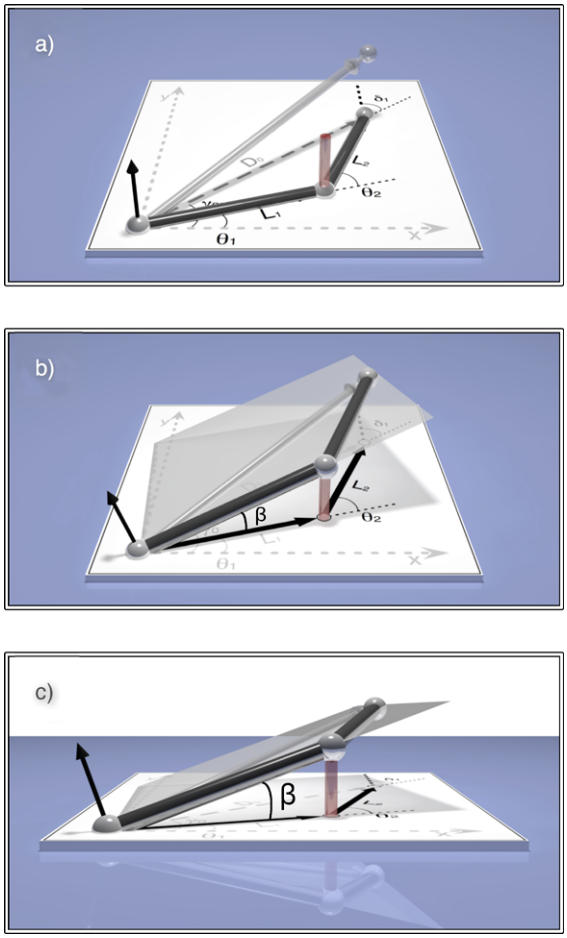

Fig. 18 Graphical representation: two segments, upper arm (L_1), lower arm (L_2), D_1

there is no disturbance introduced into the relations of rotations. Still the rotational parts of the network are in a stable state. The network will remain in the overall state because the only other solution would be to distribute the displacement onto the two other translations. But these are the segments with a fixed length. So, the network is stuck in this situation. The solution is to use the displacement which accumulates in the dual quaternions representing the joints as an error signal affecting the joint value.

How can one account for this error? We explain the solution directly for the general case, using a three dimensional example (Fig. 18), but only concentrate on one equation. One equation can be described as a triangle, a concatenation of transformations which always lie in one plane. Suppose, the current state of the variables is forming a solution, i.e., they represent a coherent configuration. This is indicated in Fig. 18(a) through the two black segments and the other transformations represented as lines on the plane. A change of the translational part of the transformation describing the endpoint of the kinematic chain—one of the diagonals or the end-effector vector—is depicted as the transparent arrow. A translational error arises in all of the rotation describing transformation which shall be compensated (red transparent bar). For a single joint this translational error has accumulated along the sequence of transformations, but during the last transformation the error is only shifted along the segment actuated by the joint and its relative position is not changing. The error can therefore be used to construct a desired position of the following segment end point (shown in Fig. 18(b) and (c)). One can now calculate a rotation which aligns the vector between current end point and desired end point. This rotation applied to the joint rotation compensates for the error. In the example, the translational error—which is the dual part of the first joint rotation dual quaternion—of the first joint is moved along the first segment (red bar in Fig. 18(b)). To account for this error, the axis of the first joint is shifted. This can be done by a rotation which can be easily described and applied as a quaternion. The rotation needed is the one which aligns the current relative segment translation and the desired segment end point position which is given as the concatenation of the segment translation and the error translation (in Fig. 18(b) and (c)). These are the two vectors delimiting the enclosed rotation angle β. The resulting quaternion $\hat{\mathbf{q}}_{\text{error}}$ representing the compensating rotation can be set-up

$$\hat{\mathbf{q}}_{\text{error}} = \begin{bmatrix} \cos\frac{\beta}{2} & 0 \\ 0 & 0 \\ -1 * \sin\frac{\beta}{2} & 0 \\ 0 & 0 \end{bmatrix}$$

and applied to the dual quaternion representing the rotation in the first joint.

$$\hat{\mathbf{r}}'_{\theta_1} = \hat{\mathbf{q}}_{\text{error}} \hat{\mathbf{r}}_{\theta_1} \overline{\hat{\mathbf{q}}^*_{\text{error}}} \tag{19}$$

This dual quaternion is now used as the value for the joint rotation and consists only of a rotation.

References

Acosta-Calderon, C., & Hu, H. (2005). Robot imitation: Body schema and body percept. *Journal of Applied Bionics and Biomechanics*, *2*(3–4), 131–148.

Aspragathos, N. A., & Dimitros, J. K. (1998). A comparative study of three methods for robot kinematics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, *28*(2), 135–145.

Atkeson, C. G., & Hollerbach, J. M. (1985). Kinematic features of unrestrained vertical arm movements. *Journal of Neurosciences*, *5*(9), 2318–2330.

Bernstein, N. A. (1967). *The co-ordination and regulation of movements*. Oxford: Pergamon Press Ltd.

Biess, A., Liebermann, D. G., & Flash, T. (2007). A computational model for redundant human three-dimensional pointing movements: Integration of independent spatial and temporal motor plans simplifies movement dynamics. *Journal of Neurosciences*, *27*(48), 13,045–13,064.

Blanke, O., Landis, T., Spinelli, L., & Seeck, M. (2004). Out-of-body experience and autoscopy of neurological origin. *Brain*, *127*(2), 243–258.

Blanke, O., Mohr, C., Michel, C. M., Pascual-Leone, A., Brugger, P., Seeck, M., Landis, T., & Thut, G. (2005). Linking out-of-body experience and self processing to mental own-body imagery at the temporoparietal junction. *Journal of Neurosciences*, *25*(3), 550–557.

Bläsing, B. (2006). Crossing large gaps: A simulation study of stick insect behaviour. *Adaptive Behaviour*, *14*(3), 265–285.

Bläsing, B., & Cruse, H. (2004). Stick insect locomotion in a complex environment: Climbing over large gaps. *The Journal of Experimental Biology*, *207*, 1273–1286.

Blohm, G., & Crawford, J. D. (2007). Computations for geometrically accurate visually guided reaching in 3-D space. *Journal of Vision*, *7*(5), 1–22.

Bockemühl, T., Troje, N., & Dürr, V. (2010). Inter-joint coupling and joint angle synergies of human catching movements. *Human Movement Science*, *29*(1), 73–93.

Bottema, O., & Roth, B. (1979). *Theoretical kinematics*. Amsterdam: North-Holland.

Botvinick, M., & Cohen, J. (1998). Rubber hands 'feel' touch that eyes see. *Nature*, *391*(6669), 756–756.

Brooks, R. A. (1991a). Intelligence without reason. In J. Myopoulos & R. Reiter (Eds.), *Proceedings of the 12th international joint conference on artificial intelligence (IJCAI-91)* (pp. 569–595). San Mateo: Morgan Kaufmann.

Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence*, *47*, 139–159.

Chasles, M. (1830). Note sur les propriétés générales du système de deux corps semblables entr'eux et placés d'une manière quelconque dans l'espace; et sur le déplacement fini ou infiniment petit d'un corps solide libre. *Bulletin des Sciences Mathematiques, Astronomiques, Physiques et Chimiques*, *14*(321–326).

Clifford, W. (1882). *Mathematical papers*. London: Macmillan.

Cothros, N., Wong, J. D., & Gribble, P. L. (2006). Are there distinct neural representations of object and limb dynamics? *Experimental Brain Research*, *173*(4), 689–697.

Cruse, H. (1979). The control of the anterior extreme position of the hindleg of a walking insect. *Physiological Entomology*, *4*, 121–124.

Cruse, H. (1986). Constraints for joint angle control of the human arm. *Biological Cybernetics*, *54*, 125–132.

Cruse, H. (1999). Feeling our body—the basis of cognition? *Evolution and Cognition*, *5*(2), 162–173.

Cruse, H. (2003). The evolution of cognition: A hypothesis. *Cognitive Science*, *27*(1), 135–155.

Cruse, H., & Brüwer, M. (1987). The human arm as a redundant manipulator: The control of path and joint angles. *Biological Cybernetics*, *57*(1–2), 137–144.

Cruse, H., & Hübner, D. (2008). Selforganizing memory: Active learning of landmarks used for navigation. *Biological Cybernetics*, *99*(3), 219–236.

Cruse, H., & Steinkühler, U. (1993). Solution of the direct and inverse kinematic problems by a common algorithm based on the mean of multiple computations. *Biological Cybernetics*, *69*, 345–351.

Daniilidis, K. (1999). Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, *18*, 286–298.

Davidson, P. R., & Wolpert, D. M. (2004). Internal models underlying grasp can be additively combined. *Experimental Brain Research*, *155*(3), 334–340.

de Vignemont, F. (2010). Body schema and body image–pros and cons. *Neuropsychologia*, *48*(3), 669–680.

Desmurget, M., & Grafton, S. (2000). Forward modeling allows feedback control for fast reaching movements. *Trends in Cognitive Sciences*, *4*(11), 423–431.

Flash, T., & Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neurosciences*, *5*(7), 1688–1703.

Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1996). *Computer graphics: Principles and practice in C* (2nd ed.). Upper Saddle River: Pearson.

Frith, C. D., Blakemore, S. J., & Wolpert, D. M. (2000). Abnormalities in the awareness and control of action. *Philosophical Transactions of the Royal Society of London: Biological Sciences*, *355*, 1771–1788.

Funda, J., & Paul, R. (1990). A computational analysis of screw transformations in robotics. *IEEE Transactions on Robotics and Automation*, *6*(3), 348–356.

Ghahramani, Z., & Wolpert, D. M. (1997). Modular decomposition in visuomotor learning. *Nature*, *386*(6623), 392–395.

Glenberg, A. M. (1997). What memory is for. *Behavioural and Brain Sciences*, *20*(1).

Govindu, V. M. (2004). Lie-algebraic averaging for globally consistent motion estimation. In *Computer vision and pattern recognition, IEEE computer society conference on* (Vol. 1, pp. 684–691).

Grush, R. (2004). The emulation theory of representation: Motor control, imagery, and perception. *Behavioural and Brain Sciences*, *27*, 377–442.

Hamilton, W. (1844). On quaternions. In *Proceedings of the Royal Irish Academy*.

Hamilton, W. (1866). *Elements of quaternions*. London: Longmans Green. New York: Chelsea, 1969.

Hanson, A. J. (2005). *The Morgan Kaufmann series in interactive 3D technology, Visualizing quaternions*. San Mateo: Morgan Kaufmann.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, *42*, 335–346.

Hartmann, G., & Wehner, R. (1995). The ant's path integration system: A neural architecture. *Biological Cybernetics*, *73*(6), 483–497.

Hesslow, G. (2002). Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, *6*(6), 242–247.

Hoffmann, M., Marques, H., Arieta, A. H., Sumioka, H., Lungarella, M., & Pfeifer, R. (2010). Body schema in robotics: A review. *IEEE Transactions on Autonomous Mental Development*, *2*(4), 304–324.

Honegger, H. W. (1981). A preliminary note on a new optomotor response in crickets: Antennal tracking of moving targets. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioural Physiology*, *142*(3), 419–421.

Imamizu, H., & Kawato, M. (2008). Neural correlates of predictive and postdictive switching mechanisms for internal models. *Journal of Neurosciences*, *28*(42), 10,751–10,765.

Jeannerod, M. (1999). To act or not to act: Perspectives on the representation of actions. *Quarterly Journal of Experimental Psychology*, *52A*, 1–29.

Kavan, L., & Žára, J. (2005). Spherical blend skinning: A real-time deformation of articulated models. In *I3D '05: Proceedings of the 2005 symposium on interactive 3D graphics and games* (pp. 9–16). New York: ACM.

Kavan, L., Collins, S., O'Sullivan, C., & Žára, J. (2006). *Dual quaternions for rigid transformation blending* (Technical report TCD-CS-2006-46). Trinity College Dublin.

Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2007). Skinning with dual quaternions. In *2007 ACM SIGGRAPH symposium on interactive 3D graphics and games* (pp. 39–46). New York: ACM Press.

Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2008). Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics*, *27*(4), 105.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, *9*, 718–727.

Kawato, M., & Gomi, H. (1992). The cerebellum and VOR/OKR learning models. *Trends in Neurosciences*, *15*(11), 445–453.
Kindermann, T., & Cruse, H. (2002). MMC—a new numerical approach to the kinematics of complex manipulators. *Mechanism and Machine Theory*, *37*(4), 375–394.
Klein Breteler, M., & Meulenbroek, R. (2006). Modeling 3D object manipulation: Synchronous single-axis joint rotations? *Experimental Brain Research*, *168*(3), 395–409.
Krakauer, J. W., Ghilardi, M. F., & Ghez, C. (1999). Independent learning of internal models for kinematic and dynamic control of reaching. *Nature Neuroscience*, *2*(11), 1026–1031.
Luenberger, D. G. (1984). *Linear and nonlinear programming*. Reading: Addison-Wesley.
Makarov, V., Song, Y., Velarde, M., Hübner, D., & Cruse, H. (2008). Elements for a general memory structure: Properties of recurrent neural networks used to form situation models. *Biological Cybernetics*, *98*(5), 371–395.
Makin, T. R., Holmes, N. P., & Ehrsson, H. H. (2008). On the other hand: Dummy hands and peripersonal space. *Behavioural Brain Research*, *191*(1), 1–10.
Mataric, M. J. (1999). Behaviour-based robotics. In R. A. Wilson & F. C. Keil (Eds.), *MIT encyclopedia of cognitive sciences* (pp. 74–77). Cambridge: MIT Press.
Mataric, M. J. (2002). Situated robotics. In *Encyclopedia of cognitive science*. London: Nature Publishing Group, Macmillan Reference Limited.
Matheson, T., & Dürr, V. (2003). Load compensation in targeted limb movements of an insect. *Journal of Experimental Biology*, *206*, 3175–3186.
Maxwell, E. A. (1951). *General homogeneous coordinates in space of three dimensions*. Cambridge: Cambridge University Press.
McCarthy, J. (1990). *Introduction to theoretical kinematics*. Cambridge: MIT Press.
McFarland, D., & Bösser, T. (1993). *Intelligent behaviour in animals and robots*. Cambridge: MIT Press.
Metzinger, T. (2006). Different conceptions of embodiment. *Psyche*, *12*(4).
Miall, R., Weir, D., Wolpert, D., & Stein, J. (1993). Is the cerebellum a Smith predictor? *Journal of Motor Behaviour*, *25*(3), 203–216.
Morasso, P. (1981). Spatial control of arm movements. *Experimental Brain Research*, *42*(2), 223–227.
Morasso, P., & Sanguineti, V. (1994). Self-organizing topographic maps and motor planning. In *SAB94: Proceedings of the third international conference on simulation of adaptive behaviour: from animals to animats* (Vol. 3, pp. 214–220). Cambridge: MIT Press.
Muller, C. M. P., Brenner, E., & Smeets, J. B. J. (2009). Maybe they are all circles: Clues and cues. *Journal of Vision*, *9*(9), 10.1-5. doi:10.1167/9.9.10.
Murray, R. M., Li, Z., & Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation*. Boca Raton: CRC.
Mussa-Ivaldi, F., Morasso, P., & Zaccaria, R. (1988). Kinematic networks distributed model for representing and regularizing motor redundancy. *Biological Cybernetics*, *60*(1), 1–16.
Niven, J. E., Buckingham, C. J., Lumley, S., Cuttle, M. F., & Laughlin, S. B. (2009). Visual targeting of forelimbs in ladder-walking locusts. *Current Biology*, *20*(1), 86–91.
Page, K. L., Zakotnik, J., Durr, V., & Matheson, T. (2008). Motor control of aimed limb movements in an insect. *Journal of Neurophysiology*, *99*(2), 484–499.
Pfeifer, R., & Scheier, C. (2001). *Understanding intelligence*. Cambridge: MIT Press.
Rosenbaum, D. A., Engelbrecht, S., Bushe, M., & Loukopoulos, L. (1993). Knowledge model for selecting and producing reaching movements. *Journal of Motor Behaviour*, *25*, 217–227.
Rosenbaum, D. A., Loukopoulos, L., Meulenbroek, R., Vaughan, J., & Engelbrecht, S. (1995). Planning reaches by evaluating stored postures. *Psychological Review*, *102*, 28–67.
Rosenbaum, D. A., Meulenbroek, R. J., Vaughan, J., & Jansen, C. (2001). Posture-based motion planning: applications to grasping. *Psychological Review*, *108*(4), 709–734.
Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations (parallel distributed processing)*. Cambridge: MIT Press.
Schilling, M. (2009). Dynamic equations in MMC networks: Construction of a dynamic body model. In *Proc. of the 12th international conference on climbing and walking robots and the support technologies for mobile machines (CLAWAR)*.
Schilling, M., & Cruse, H. (2007). Hierarchical MMC networks as a manipulable body model. In *Proceedings of the international joint conference on neural networks (IJCNN 2007)*, Orlando, FL (pp. 2141–2146).
Schilling, M., & Cruse, H. (2008). The evolution of cognition—from first order to second order embodiment. In I. Wachsmuth & G. Knoblich (Eds.), *Modeling communication with robots and virtual humans*. Berlin: Springer.
Schmitz, J., Schneider, A., Schilling, M., & Cruse, H. (2008). No need for a body model: Positive velocity feedback for the control of an 18-dof robot walker. *Applied Bionics and Biomechanics*, Special Issue on Biologically Inspired Robots, *5*(3), 135–147.
Shadmehr, R., & Mussa-Ivaldi, F. (1994). Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience*, *14*, 3208–3224.
Shaw, B. (1903). *Man and superman: A comedy and a philosophy*. London.
Shoemake, K. (1985). Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on computer graphics and interactive techniques* (pp. 245–254). New York: ACM Press.
Smeets, J. B. J., van den Dobbelsteen, J. J., de Grave, D. D. J., van Beers, R. J., & Brenner, E. (2006). Sensory integration does not lead to sensory calibration. *Proceedings of the National Academy of Sciences of the United States of America*, *103*(49), 18,781–18,786. doi:10.1073/pnas.0607687103.
Soechting, J., Buneo, C., Herrmann, U., & Flanders, M. (1995). Moving effortlessly in three dimensions: Does Donders' law apply to arm movement? *Journal of Neurosciences*, *15*(9), 6271–6280.
Steels, L. (2003). Intelligence with representation. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, *361*(1811), 2381–2395.
Steinkühler, U. (1994). *Mmc-modelle zur lösung kinematischer aufgabenstellungen eines redundanten manipulators*. Ph.D. thesis, University of Bielefeld.
Steinkühler, U., & Cruse, H. (1998). A holistic model for an internal representation to control the movement of a manipulator with redundant degrees of freedom. *Biological Cybernetics*, *79*(6), 457–466.
Strauss, R., & Pichler, J. (1998). Persistence of orientation toward a temporarily invisible landmark in drosophila melanogaster. *Journal of Comparative Physiology A*, *182*, 411–423.
Stringer, S., & Rolls, E. (2007). Hierarchical dynamical models of motor function. *Neurocomputing*, *70*, 975–990.
Uno, Y., Kawato, M., & Suzuki, R. (1989). Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, *61*(2), 89–101.
van Beers, R., Wolpert, D., & Haggard, P. (2002). When feeling is more important than seeing in sensorimotor adaptation. *Current Biology*, *12*, 834–837.
Wang, L. C. T., & Chen, C. C. (1991). A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, *7*(4), 489–499.
Wang, X. (1999). Three-dimensional kinematic analysis of influence of hand orientation and joint limits on the control of arm postures and movements. *Biological Cybernetics*, *80*(6), 449–463.

Webb, B. (2004). Neural mechanisms for prediction: Do insects have forward models? *Trends in Neurosciences*, *27*(5), 278–282.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, *10*, 47–53.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, *9*(4), 625–636.

Wolpert, D., Ghahramani, Z., & Jordan, M. (1995). An internal model for sensorimotor integration. *Science*, *269*, 1880–1882.

Wolpert, D., Miall, R., & Kawato, M. (1998). Internal models in the cerebellum. *Trends in Cognitive Sciences*, *2*(9), 338–347.

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*(7–8), 1317–1329.

Yang, A., & Freudenstein, F. (1964). Application of dual-number quaternion algebra to the analysis of spatial mechanisms. *ASME Journal of Applied Mechanics*, 300–308.

Yoshikawa, T. (1985). Manipulability and redundancy control of robotic mechanisms. In *Proceedings of the IEEE int. conference on robotics and automation*, St. Louis, Missouri (pp. 1004–1009).

Malte Schilling is a PostDoc at the International Computer Science Institute (ICSI) in Berkeley. His work concentrates on internal models, their grounding in behaviour and their application in higher-level cognitive function like planning ahead or communication. Before moving to Berkeley, he worked at the Sony Computer Science Laboratory in Paris on the embodiment of language and the connections between internal models and language.

He received his PhD in Biology at the University of Bielefeld in January, 2010. His PhD project focuses on the control of a hexapod robot, applying insights from biological experiments with stick insects on a reactive control model, and enhancing this control through cognitive capabilities by introducing an internal body model implemented as a neuronal network. After finishing his PhD he became a Responsible Investigator at the Center of 'Excellence Cognitive Interaction Technology' (CITEC), University of Bielefeld. He has studied Computer Science (artificial intelligence as a main subject) at the University of Bielefeld and finished the Diploma with his thesis on knowledgebased systems in the context of virtual environments.