Robotic Assembly Using a Singularity-Free Orientation Representation Based on Quaternions

Stolt, Andreas; Linderoth, Magnus; Robertsson, Anders; Johansson, Rolf

Published in:
10th IFAC Symposium on Robot Control

DOI:
10.3182/20120905-3-HR-2030.00074

2012

Robotic Assembly Using a Singularity-Free Orientation Representation Based on Quaternions ⋆

Andreas Stolt * **Magnus Linderoth** * **Anders Robertsson** *
Rolf Johansson *

* *Department of Automatic Control, LTH, Lund University, Sweden*
(corresponding author e-mail: andreas.stolt@control.lth.se).

Abstract: New robotic applications often require physical interaction between the robot and its environment. To this purpose, external sensors might be needed, as well as a suitable way to specify the tasks. One complication that might cause problems in the task execution is orientation representation singularities. In this paper quaternions are used as a singularity-free orientation representation within the constraint-based task specification framework. The approach is experimentally verified in a force controlled assembly task. The task chosen contains a redundant degree of freedom that is exploited using the constraint-based task specification framework.

Keywords: Robot control, force control, assembly, industrial robots

1. INTRODUCTION

Industrial robots are traditionally only position controlled, and the tasks are given as trajectories to follow. New applications, however, increase the need for integration of external sensing from the workspace, such as vision and force. A framework that can be used for incorporation of sensors and general task specification is the constraint-based task specification framework (De Schutter et al., 2007), or simply *iTaSC* (instantaneous Task Specification using Constraints). Here the robot motion is specified by imposing constraints, e.g., force, velocity, or position constraints.

The goal of the current work is to create a framework for sensor-guided assembly that should be possible to use for non-expert robot operators. Previous work in this direction by the authors is presented in (Stolt et al., 2011), where a snapfit assembly is considered together with how uncertainties could be resolved and how learning strategies could be used to increase the assembly speed. The framework used is based on iTaSC, where Euler angles commonly are used as a representation for orientation. A main reason for using this representation is that it is intuitive to work with and that it offers a minimial representation for orientations, unfortunately it has problems with representation singularitites.

One way to avoid the problems with an Euler angle representation, but keeping the intuitive orientation description, is to use an internal singularity-free representation. Two such representations are quaternions and rotation matrices. The former representation is the one considered in this paper, due to the fact that a quaternion only needs 4 parameters and a rotation matrix needs 9 (not all unique). Another possibility is to switch between different Euler angle representations that have the representation singularity in different orientations, as described in (Singla et al., 2005). A drawback with a switching solution is that it would be cumbersome to handle all possible Euler angle parametrizations, if the user should have the possibility to choose from all of them. The mapping of constraints between the switching representations might also be a problem. An advantage with the quaternion representation is that it always gives orthogonal rotation axes, which is not the case for an Euler angle representation. If the task requires constraints on non-orthogonal rotation axes, the remedy is to use more than one kinematic chain.

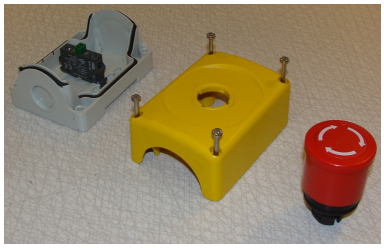

Fig. 1. The emergency stop button used as an experimental case.

A general method to handle non-minimal representations, such as quaternions, within iTaSC is suggested in (De Schutter et al., 2007). Both this approach and the one presented in this paper use the fact that there exists a minimal representation on the velocity level. In fact, they can be proven to be equal. This paper further presents how the quaternion representation is integrated into the iTaSC methodology, making it possible for the user to specify tasks without having to worry about orientation representation singularities. The approach is experimentally verified in an implementation of a force-controlled assembly task. The task considered is a subassembly of an emergency stop button; see Fig. 1 for an overview of the involved parts. The red button should be inserted into the yellow case, i.e., a peg-in-hole assembly. The button is assumed to be rotationally symmetric, which introduces a redundant degree of freedom in the task. This redundancy is exploited using the iTaSC framework.

⋆ The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 – Challenge 2 – Cognitive Systems, Interaction, Robotics – under grant agreement No 230902 - ROSETTA. This document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained herein.

Quaternions have previously been used in many contexts, such as robotics, computer graphics, and attitude control of aircrafts and spacecrafts. A primary reason for working with quaternions is that they are a singularity-free orientation representation. In (Antonelli and Chiaverini, 1998) an underwater vehicle-manipulator system is controlled in a singularity-free manner using the unit quaternion, and the quaternion is also used in (Caccavale and Siciliano, 2001) to avoid representation singularities in control of a redundant manipulator on a spacecraft. In (Wen and Kreutz-Delgado, 1991) the unit quaternion is used as a singularity-free orientation representation to derive a large family of globally stable control laws for the attitude control problem. Another example where the quaternion is used to avoid singularities is (Xian et al., 2004), where it is used to implement a task-space tracking control scheme.

Quaternions have also been used in connection with SLAM, e.g., in (Kyrki, 2008) they are used to represent similarity transforms. A general framework on how to handle redundancy is presented in (Rocco and Zanchettin, 2010). Previous work in robotic assembly can e.g. be found in (Arai et al., 2006), where optimization of force control parameters with respect to cycle time was made in assembly of a clutch. An example from the automotive industry is (Gravel et al., 2008), which describes powertrain assembly. In (Zhang et al., 2005) synchronized Petri nets were used to model the assembly process and an experimental evaluation was made with a peg-in-hole assembly.

2. PRELIMINARIES

2.1 Constraint-based task specification

A thorough explanation of the constraint-based task specification methodology, iTaSC, is given in (De Schutter et al., 2007). A summary of the parts that are relevant for this paper is given below.

The iTaSC framework specifies the relative motion of objects by imposing constraints. A kinematic chain, consisting of two object frames and two feature frames, are used to specify the constraints. The object frames are attached on the objects to be manipulated and on the robot, and the feature frames are attached to relevant features for the task. The frames should be defined such that the task becomes as easy as possible to specify. A kinematic chain should have 6 degrees of freedom, denoted by χ_f, the *feature coordinates*. They are distributed over the transformations between the object and the feature frames.

There might exist uncertainties in the pose between the previously defined coordinate frames. To model these uncertainties an extra transformation between each of the object and feature frames are introduced. The degrees of freedom in these transformations are denoted by χ_u, the *uncertainty coordinates*.

The variables to be constrained are chosen by specifying outputs y. In general, each output can be a function of the feature and the robot joint coordinates, but with properly chosen kinematic chains, the outputs will in most cases directly correspond to a subset of the feature coordinates.

The iTaSC framework is suitable to handle both over- and under-constrained tasks, as well as manipulators with redundant degrees of freedom. For instance, in the velocity based control scheme of iTaSC the redundancy can be used to optimize some criterion of the joint velocities. In case of an over-constrained task, weighting or prioritizing of the constraints is used to calculate the desired motion.

2.2 Orientation representation

Orientation in the kinematic chain is usually represented by Euler angles, i.e., three consecutive rotations around given coordinate axes. The reason for choosing Euler angles is that they are intuitive to work with and easy to specify in a kinematic chain. Furthermore, they offer a convenient way of parametrizing an orientation, and also make it possible to control all three angles separately.

There are, however, several problems with an Euler angle representation. The first one is that the parameterization is not unique, e.g. $(\frac{\pi}{2}, \frac{\pi}{2}, 0)$ and $(-\frac{\pi}{2}, -\frac{\pi}{2}, \pi)$ are two examples of ZYZ-Euler angles that represent the same orientation. This results in problems when the inverse kinematics problem is considered, i.e., when calculating the Euler angles for a given orientation. Another problem is the inherent representation singularity, which occurs when two rotation axes are aligned. This results in the Jacobian of the kinematic chain losing rank. The iTaSC motion solver uses an inverse of this Jacobian, and a representation singularity is therefore highly inconvenient.

2.3 Quaternions

Quaternions (Hamilton, 1840) are an extension of the complex number system. A quaternion Q consists of a scalar part, $Q_s \in \mathbb{R}$, and a vector part, $\bar{Q}_v \in \mathbb{R}^3$, according to

$$Q = \left(Q_s,\, \bar{Q}_v \right) \tag{1}$$

Unit quaternions are a suitable choice as a representation for rotations. The rotation around an axis $\bar{v}$, $|\bar{v}| = 1$, with the angle θ is then given by the quaternion

$$Q = \left(\cos\left(\theta/2\right),\, \sin\left(\theta/2\right) \bar{v} \right) \tag{2}$$

The use of quaternions for representing rotations does not exhibit the problems of the Euler angle representation. The drawback is, however, the non-minimality of the quaternion representation. Four parameters are needed together with the normalization constraint, $||Q|| = 1$. The intuitivity of the Euler angles is also lost.

3. QUATERNION REPRESENTATION

3.1 Kinematics

To be able to incorporate quaternions in a kinematic chain in the iTaSC framework, a new rotation transformation has to be introduced. It is a general 3D rotation, and its current value is described by a quaternion. One difference from other types of transformations previously used within iTaSC is that this has three degrees of freedom, and the corresponding feature coordinate is a 4D-vector. A kinematic chain can contain several of these quaternion transformations, but only one can be part of the feature coordinates. The others might be used to introduce uncertain and constant reorientations.

When formulating the motion specification with iTaSC all constraints are transformed to velocity constraints. Force and position constraints are handled by the use of controllers that output a desired velocity to achieve the constraints. The velocity of an ordinary feature coordinate is simply its time derivative, but for a quaternion transformation this is not the

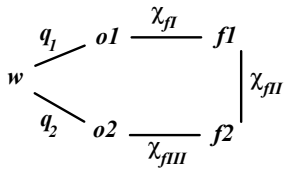

Fig. 2. The inverse kinematics problem is solved by considering the position loop constraint that is defined by the kinematic chain, w denoting the world coordinate frame, q_1 and q_2 robot joint coordinates, $o1$ and $o2$ object frames, $f1$ and $f2$ feature frames, and $\chi_f = (\chi_{fI}, \chi_{fII}, \chi_{fIII})$ the feature coordinates.

desired way to describe a velocity constraint, as the derivative of the quaternion parameters are even less intuitive than the quaternion itself. Therefore a geometric approach is adopted, and the velocity considered is the angular velocity in the local coordinate system described by the quaternion. This means that the angular velocities around the coordinate axes described by the quaternion are considered as feature coordinates and not the quaternion parameters. The quaternion is used to keep track of the current orientation. Using the angular velocities as feature coordinates makes it easy to calculate the part of the Jacobian of the kinematic chain belonging to the quaternion transformation. It is given in (3), where x_i, y_i, and z_i are the rotation axes, i.e., the coordinate axes in the coordinate system described by the quaternion. The vector $\bar{t}$ represents the vector from the rotation point to the endpoint of the kinematic chain.

$$J_Q = \begin{bmatrix} x_i \times \bar{t} & y_i \times \bar{t} & z_i \times \bar{t} \\ x_i & y_i & z_i \end{bmatrix} \tag{3}$$

Inverse kinematics, i.e., the problem of finding the feature coordinates when the rest of the kinematic chain is known, can be solved in a similar way as the model update proposed in (De Schutter et al., 2007). Consider the position loop constraint (4) represented as a product of homogenous transformation matrices [1] , see also the illustration in Fig. 2.

$$\begin{aligned} T_{w\to o1}(q_1)T_{o1\to f1}(\chi_{fI})T_{f1\to f2}(\chi_{fII})... \\ T_{f2\to o2}(\chi_{fIII})T_{o2\to w}(q_2) = I_{4\times 4} \end{aligned} \tag{4}$$

When there is a quaternion Q in the kinematic chain, the feature coordinates can be written $\chi_f = (Q, \bar{\chi}_f)$, where $\bar{\chi}_f$ contains all feature coordinates except for Q. An equivalent formulation of (4) is (5), where the fact that q_1 and q_2 are known and constant has been used.

$$T_1T_2(\bar{\chi}_f)T_3(Q)T_4(\bar{\chi}_f)T_5 = I_{4\times 4} \tag{5}$$

This position loop constraint can be solved for $\bar{\chi}_f$ and Q in an iterative fashion. By first assuming that $\bar{\chi}_f$ is known, it is possible to calculate Q such that the orientation part of (5) is fulfilled, i.e., only the rotation matrix part of the homogenous transformations is considered. To then solve (5) for $\bar{\chi}_f$ one can for instance make a linear approximation. Let us denote the left-hand side of (5) for $T(\bar{\chi}_f)$ and the current estimate of $\bar{\chi}_f$ for $\hat{\bar{\chi}}_f$, then (6) holds, where R_{err} is the orientation error represented by a rotation matrix and t_{err} the translation error represented by a Cartesian vector.

$$T(\hat{\bar{\chi}}_f) = T_{err}I_{4\times 4} \ , \ T_{err} = \begin{bmatrix} R_{err} & t_{err} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{6}$$

[1] The uncertainty coordinates have been omitted here, but the inclusion of them are straightforward as they are considered to be known and constant in these calculations.

A linear approximation to describe $\Delta\bar{\chi}_f = \bar{\chi}_f - \hat{\bar{\chi}}_f$ is (7), where a_{err} is an axis/angle representation of R_{err} and $J_{\bar{\chi}_f}$ the Jacobian for $T(\bar{\chi}_f)$.

$$J_{\bar{\chi}_f}\Delta\bar{\chi}_f = \begin{bmatrix} t_{err} \\ a_{err} \end{bmatrix} \tag{7}$$

Normally there are 6 feature coordinates, but as $\bar{\chi}_f$ does not contain the quaternion only 3 feature coordinates remain. This means that a least-squares solution can be used to solve (7) for $\Delta\bar{\chi}_f$ and update $\hat{\bar{\chi}}_f$.

Iteration of the described procedure is performed until the error is small enough, i.e., the right-hand side of (7). As the inverse kinematics is calculated continuously during operation and the coordinate values in the previous sample are used as starting values in the next sample, usually only one or a few iterations are needed. The only exception is in the start-up of the program, when the initial guess might be far off.

3.2 Euler angle references

The value of the quaternion transformation is possible to specify in any format, and it is possible to give a desired orientation in Euler angles. This is no problem as the transformation this way is unique, i.e., all possible Euler angle coordinates for a particular orientation result in the same quaternion. The same holds for velocity and torque references. In the velocity control case the desired Euler angle velocity is transformed to a desired angular velocity using the Jacobian, relating the Euler angle time derivatives to the angular velocity. When a torque constraint is active the controller output is the desired velocity, which is transformed using the Jacobian, in the same way as the velocity control case.

3.3 Hybrid control

Controlling a quaternion in a kinematic chain to a desired value is fairly straightforward. If the current orientation is denoted Q_{cur} and the desired orientation Q_{des}, then the orientation error is given by $Q_{err} = Q_{cur}^{-1} * Q_{des}$, where $*$ denotes quaternion multiplication. By exploiting the fact that the error describes a rotation, its rotation axis $\bar{v}_{err}$ and its rotation angle θ_{err} can be calculated from the parametrization (2). The orientation error can now be eliminated by applying the desired angular velocity (8), where K is a gain factor.

$$\bar{\omega}_{des} = K\theta_{err}\bar{v}_{err} \tag{8}$$

It is a bit more difficult to apply hybrid control, e.g., when it is desired to control the orientation around one axis and torque around another. A solution to this problem is to continuously update the quaternion reference, by integrating the velocity references given by the torque controller. The position (orientation) controller is constrained to only apply velocity corrections around the axes that are position controlled, i.e., the desired velocity from the controller is projected onto the axes that are position controlled. Updating the reference for the part of the orientation that is position controlled requires a complete orientation reference. Only giving the Euler angles for the position controlled directions is not enough, as the complete orientation description can not be uniquely calculated from this information.

The integration of the quaternion reference is made by applying a rotation with constant angular velocity during one sample

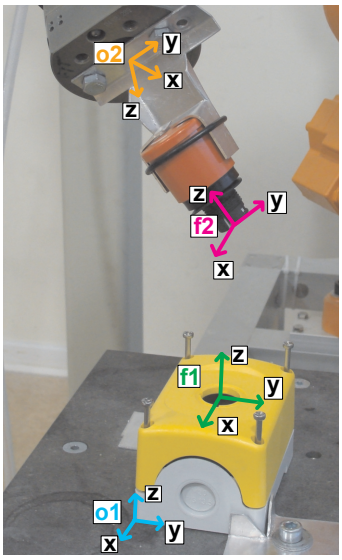

Fig. 3. Illustration of the different coordinate frames used in the assembly task.

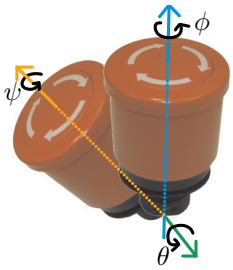

Fig. 4. Illustration of the Euler angle representation (ZYZ) of the orientation of the button. ϕ is a rotation around the z-axis, θ is then a rotation around the y-axis in the new coordinate system defined by the first rotation. Finally ψ is a rotation around the new z-axis.

period, i.e., multiplying with the quaternion (9), where $\bar{\omega}$ is the angular velocity and h the sample period.

$$Q_{int} = (\cos(|\bar{\omega}|h/2),\ \sin(|\bar{\omega}|h/2)\bar{\omega}/|\bar{\omega}|) \tag{9}$$

3.4 Redundancy

A task may not need all available degrees of freedom, and this redundancy should be exploited. If the orientation that is described by a quaternion is part of the redundancy it can be handled by not specifying the velocity around the redundant axis. When the quaternion is position controlled the calculated desired velocity should be projected onto the rotation axes that are part of the task. This means that any desired quaternion can be specified, but that only errors projected into the degrees of freedom that are part of the task will be eliminated.

Introducing quaternions do not alter the way iTaSC handles the redundancy, as the quaternion on the velocity level is completely described by three angular velocities.

4. ASSEMBLY SCENARIO

The assembly scenario used to illustrate the quaternion approach is a part of the assembly of an emergency stop button, see Fig. 3. The red button should be placed in the hole in the yellow case, i.e., a peg-in-hole assembly. The button is assumed to be rotationally symmetric, although this is not exactly true. Making this assumption, however, makes the task redundant, as the rotation around the symmetry axis does not matter. If the button is grasped in such a way that the symmetry axis coincides with the last joint axis of the robot, the redundancy is trivial and only results in the position of the last robot joint being unconstrained. The gripper is constructed in such a way that the redundant degree of freedom is not trivial. A wrist-mounted 6 degrees-of-freedom force/torque sensor is used to perform the assembly.

4.1 Kinematic chain

One kinematic chain is used in the assembly task and the object and feature frames related to it are shown in Fig. 3.

- Object frame $o1$ is attached to the box. It is related to the world coordinate frame by a fix transformation.
- Feature frame $f1$ has its origin in the center of the hole on top of the yellow case. The orientation is the same as $o1$.
- Feature frame $f2$ is attached to the endpoint of the button and its orientation is illustrated in Fig. 3.
- Object frame $o2$ has its origin on the base of the gripper and the same orientation as the robot flange frame. It is related to $f2$ by a fix transformation.

The feature coordinates χ_f are divided into three groups depending on which frames they relate to, according to

$$\begin{array}{ll} \chi_{fI} = (-) & o1 \rightarrow f1 \\ \chi_{fII} = (x, y, z, Q) & f1 \rightarrow f2 \\ \chi_{fIII} = (-) & f2 \rightarrow o2 \end{array}$$

The first three feature coordinates, (x, y, z), are translations along the coordinate axes of $f1$. Then it is intuitive to describe the orientation of the red button with ZYZ-Euler angles, illustrated in Fig. 4. First a rotation around the z-axis is made, and then a rotation around the y-axis of the new coordinate system. Finally a rotation around the new z-axis is introduced. The last axis is the symmetry axis of the button and this rotation thus corresponds to the redundancy in the task. This Euler angle representation would certainly cause trouble if it would have been used for feedback control. The reason is that it has a representation singularity close to the target position, because in this position the second angle coordinate is zero and the first and the third rotation axes coincide. Instead, the three Euler angles are represented by a quaternion, Q, internally.

Outputs are chosen as Eq. (10), where the two last outputs are specified on the velocity level and correspond to the quaternion (ω_x and ω_y denote the angular velocities around the x- and the y-axis, respectively, in the coordinate frame described by the quaternion, i.e., frame $f2$). The actual values of y_4 and y_5 are not defined, but the torque corresponding to these outputs is the torque around the corresponding rotation axes. The last angular velocity, ω_z, is not chosen as output as it will not be constrained in any way, due to the assumption of rotational symmetry.

$$y_1 = x\ ,\quad y_2 = y\ ,\quad y_3 = z\ ,\quad \dot{y}_4 = \omega_x\ ,\quad \dot{y}_5 = \omega_y \tag{10}$$

Uncertainties in the task include the exact location and orientation of the box, and the grasp of the button. They are, however, resolved using guarded search motions, i.e., the motion is velocity controlled in the search direction and stopped when a contact force is detected. Once contact is made, it is maintained by using force control, and hence no explicit uncertainty coordinates are used to model this uncertainty.

4.2 Redundancy

The iTaSC motion specification is calculated by solving for the robot joint velocities, $\dot{q}$, in (11). The matrix A relates $\dot{q}$ to the desired output velocities $\dot{y}_d^0$, see (De Schutter et al., 2007) for details.

$$A\dot{q} = \dot{y}_d^0 \tag{11}$$

When the task is redundant, or over-constrained, the matrix A will not be square, and hence a pseudoinverse must be used. In case of a redundant task the weighted pseudoinverse $A^\dagger$ in (12) can be used. The interpretation is that the optimization problem (13) is solved, where M is a weighting matrix.

$$A^{\dagger} = M^{-1}A^{T}\left(AM^{-1}A^{T}\right)^{-1} \tag{12}$$

$$\begin{array}{ll} \text{minimize (over } \dot{q}) & \dot{q}^{T}M\dot{q} \\ \text{subject to} & \dot{y}_{d}^{0} = A\dot{q} \end{array} \tag{13}$$

4.3 Assembly strategy

The assembly strategy is designed in such a way that the uncertainties are resolved during execution of the task. The position of the yellow case is assumed not to be known well enough for hitting the hole with the button in the upright position. But the uncertainty is small enough for an approach with a tilted button to hit the hole (Fig. 3). Once the hole is found, force control is used to find the center of it. Then the button is reoriented towards what is assumed to be the upright position while using force control to press downwards, such that the button gradually slides down into the hole. Torque control is then used to find the correct orientation. This strategy is modeled with a state machine, where each state has the following actions:

(1) Goto start position
(2) Search for contact in z-direction (output y_3)
(3) Force control to center of hole
(4) Reorient to the approximate upright position
(5) Control torques to zero
(6) (Release button and) move robot away

Position or force measurements are used to trigger transitions between subsequent states.

5. EXPERIMENTAL RESULTS

Force data from an experimental execution is given in Fig. 5, together with the corresponding state in the assembly sequence. The first state shown, state 2, is the search in the z-direction. The transition condition to the next state is that a large z-force is detected, and this happens at $t = 2.9$ [s]. State 3 is a search for the middle of the hole, which is made by controlling the x- and y-forces to zero while keeping a positive force in the z-direction; the reference is set to 10 [N]. The next step is then a position control of the orientation to the assumed upright position of the button, while pressing in the z-direction such that the button slides down completely into the hole. In state number 5 the torques around the x- and y-axes are controlled to zero, such that the button is completely pushed down into the hole. The last state is that the robot is moved away in the positive z-direction.

The feature coordinates with Euler angles in the kinematic chain have been calculated for the experimental execution shown in Fig. 5, and these angles are shown in Fig. 6. The initial position chosen was $\phi = -90^{\circ}$, $\theta = 36^{\circ}$ and $\psi = 90^{\circ}$, where the ψ-angle corresponds to the redundant degree of freedom. As the θ-coordinate was decreased, corresponding to the button being moved towards the upright position, the current pose was getting closer and closer to the singular configuration. When it came close, the ϕ- and the ψ-angles immediately changed with about 180°, and the reason it stopped there is probably that the singularity was only closely passed by. If this kinematic chain would have been used for control it would be hard to predict what would happen close to the singularity. Furthermore, if the program would have survived the singularity, problems might have occurred because the ϕ-coordinate had drifted away.

The redundancy in the task was handled by choosing the weighting matrix M as (14), i.e., such that it was desired to rather move the wrist of the robot (joint 4, 5 and 6) than the three first joints.

$$M = \text{diag}\,(\,10\;\;10\;\;10\;\;1\;\;1\;\;1\,) \tag{14}$$

The resulting measured redundant angular velocity from the experimental execution is shown in Fig. 7. Quite large rotations can be seen, indicating that the redundancy was exploited, especially around $t = 9$ [s] when the button is closing in on the upright position and slides down into the hole.

6. DISCUSSION

The described choice of Euler angles in the assembly task had a representation singularity close to the target position, and would therefore cause trouble in the execution of the task. The previous solution to this problem was to be careful and choose a safe orientation representation, as e.g. was done in (Stolt et al., 2011). An Euler XYZ representation would for instance be fine for the assembly scenario in this paper. Using the assembly framework and the quaternion representation described, however, relieves the user from doing these considerations, and the user only has to come up with a suitable representation for the task. In this way it is a step towards making it easier for unexperienced robot programmers to accomplish assembly tasks.

A drawback of the approach with an internal singularity-free representation is that it is not possible to use feedback from the individual Euler angles. An example is when one Euler angle direction is torque controlled and the other two position controlled, as then a reference change for the position controlled angles might be impossible to translate to the internal representation because of the Euler angle ambiguity. These situations are rare, and most scenarios relevant in assembly are not subject to this problem. In case they do occur, the remedy is to specify the complete orientation.

The current experimental implementation of the proposed quaternion representation is not completely automatic, i.e., it is not possible for a user to make the modeling and design the state machine describing the task with Euler angles and get an implementation based on the quaternion representation. The current procedure requires some manual steps, which in a real application have to be made automatic. This is, however, a shortcoming of the implementation, not of the proposed method.

The redundancy resolution should be chosen such that some criterion is optimized, e.g., the energy spent during the task execution could be minimized. The weighting matrix M in the assembly task described in this paper has been chosen quite arbitrarily, where the only objective was to show the concept of relative weighting. Further efforts should be spent to find a meaningful criterion (13).

The button was assumed to be rotationally symmetric, which is not true in reality. There are some edges that might get stuck during the insertion in the hole. This sometimes happened during execution of the assembly. Some extra oscillations are then induced, but otherwise the assembly sequence still works fine.

The force control parameters used in the assembly sequence were tuned manually and were therefore probably not optimal.

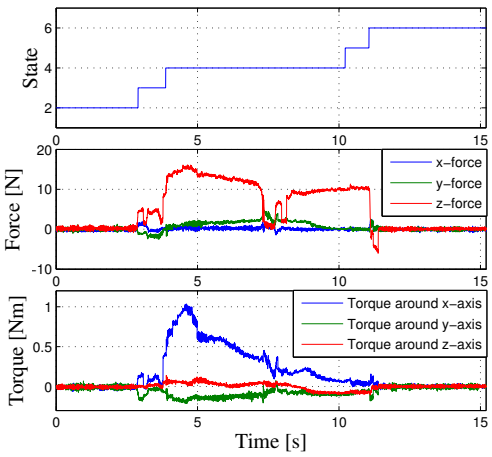

Fig. 5. Force data from an assembly sequence vs. time. The uppermost diagram shows the state sequence, the middle the forces (along the first three feature coordinate directions, i.e, they are given in frame $f1$) and the lowermost the torques (around the coordinate axes defined by the quaternion, i.e., they are given in frame $f2$).

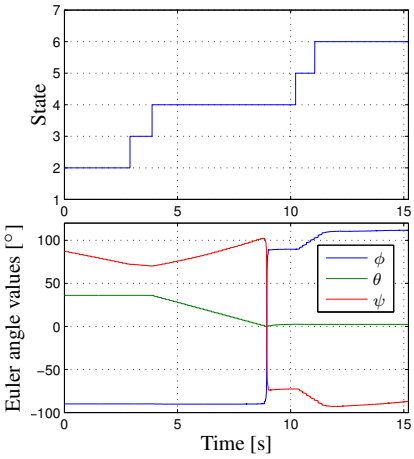

Fig. 6. Calculated Euler angles from the assembly sequence. ϕ is the first rotation around the z-axis, θ the rotation around the y-axis, and ψ the final rotation around the z-axis. Problems occur when the singular position is entered just before $t = 9$ [s].

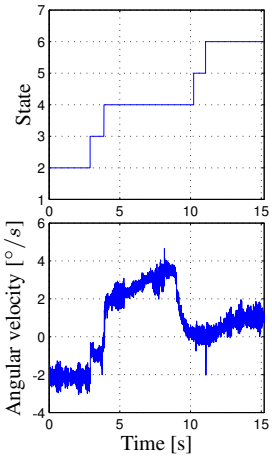

Fig. 7. Measured angular velocity in the redundant direction.

Tuning these parameters is a tedious work, and the goal is to make this process adaptive. This will further on make the implementation robust to changing environments, and also make it easier for users to specify tasks, when they do not have to be concerned about the choice of controller parameters, assuming that the adaptive laws have been designed properly.

The current assembly speed is not that fast, but very little effort has been spent on optimizing it. This is, however, something that would be important in an industrial application. If it is assumed that the same assembly is performed several times, learning approaches can be used to generate feed-forward data that can speed up the assembly. Learning can also be used to make changes in the assembly sequence, if this is appropriate.

7. CONCLUSIONS

A method to introduce a singularity-free orientation representation based on quaternions within the iTaSC framework has been described. The proposed method makes it possible to model tasks with Euler angles, and execute them such that the inherent representation singularities cause no problems. The method has further on been implemented on an industrial robot system and experimentally verified in a force controlled assembly task. The chosen task contained a redundant degree of freedom that was exploited using the iTaSC framework.

REFERENCES

Antonelli, G. and Chiaverini, S. (1998). Singularity-free regulation of underwater vehicle-manipulator systems. In *Proc. American Control Conf. (ACC)*, volume 1, 399–403. Philadelphia, USA.

Arai, T., Yamanobe, N., Maeda, Y., Fujii, H., Kato, T., and Sato, T. (2006). Increasing Efficiency of Force-Controlled Robotic Assembly -Design of Damping Control Parameters Considering Cycle Time. *CIRP Annals-Manufacturing Technology*, 55(1), 7–10.

Caccavale, F. and Siciliano, B. (2001). Quaternion-based kinematic control of redundant spacecraft/manipulator systems. In *Proc. Int. Conf. Robotics and Automation (ICRA)*, volume 1, 435–440. Seoul, Korea.

De Schutter, J., De Laet, T., Rutgeerts, J., Decré, W., Smits, R., Aertbeliën, E., Claes, K., and Bruyninckx, H. (2007). Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *Int. J. Robotics Research*, 26(5), 433.

Gravel, D., Maslar, F., Zhang, G., Nidamarthi, S., Chen, H., and Fuhlbrigge, T. (2008). Toward robotizing powertrain assembly. In *7th World Congress Intelligent Control and Automation (WCICA)*, 541–546. Chongqing, China.

Hamilton, W. (1840). On a New Species of Imaginary Quantities, Connected with the Theory of Quaternions. In *Proc. Royal Irish Academy*, volume 2, 424–434.

Kyrki, V. (2008). Quaternion representation for similarity transformations in visual SLAM. In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, 2498–2503. Nice, France.

Rocco, P. and Zanchettin, A. (2010). General parameterization of holonomic kinematic inversion algorithms for redundant manipulators. In *Proc. Int. Conf. Robotics and Automation (ICRA)*, 3721–3726. Anchorage, USA.

Singla, P., Mortari, D., and Junkins, J.L. (2005). How to avoid singularity when using Euler angles? *Advances In The Astronautical Sciences*, 119(II), 1409–1426.

Stolt, A., Linderoth, M., Robertsson, A., and Johansson, R. (2011). Force Controlled Assembly of Emergency Stop Button. In *Proc. Int. Conf. Robotics and Automation (ICRA)*, 3751–3756. Shanghai, China.

Wen, J. and Kreutz-Delgado, K. (1991). The attitude control problem. *Automatic Control, IEEE Transactions on*, 36(10), 1148–1162.

Xian, B., de Queiroz, M., Dawson, D., and Walker, I. (2004). Task-space tracking control of robot manipulators via quaternion feedback. *Robotics and Automation, IEEE Trans.*, 20(1), 160–167.

Zhang, W., Mao, T., and Yang, R. (2005). A new robotic assembly modeling and trajectory planning method using synchronized Petri nets. *The Int. J. Advanced Manufacturing Techn.*, 26(4), 420–426.