Experimental Results for Set-based Control within the Singularity-robust Multiple Task-priority Inverse Kinematics Framework

Signe Moe1, Gianluca Antonelli2, Kristin Y. Pettersen1 and Johannes Schrimpf3

Abstract—Inverse kinematics algorithms are commonly used in robotic systems to achieve desired behavior, and several methods exist to ensure the achievement of numerous tasks simultaneously. The multiple task-priority inverse kinematics framework allows a consideration of tasks in a prioritized order by projecting task velocities through the null-spaces of higher priority tasks. Recent results have extended this framework from equality tasks to also handling set-based tasks, i.e. tasks that have an interval of valid values. The purpose of this paper is to further investigate and experimentally validate this algorithm and its properties. In particular, this paper presents experimental results where a number of both set-based and equality tasks have been implemented on the 6 Degree of Freedom UR5 which is an industrial robotic arm from Universal Robots. The experiments validate the theoretical results.

I. INTRODUCTION

Robotic systems with a large number of Degrees of Freedom (DOFs) are frequently used for industrial purposes [1] and are becoming increasingly important within a variety of fields, including unmanned vehicles such as underwater [2] and aerial [3] systems.

Traditionally, robotic systems are controlled in their joint space. However, the tasks they are required to execute are often defined in the operational space, for instance given by the desired end effector position or orientation. On a kinematic level, the most common approach is to use a Jacobian-based method as a mapping from operational to joint space [4]-[6]. In particular, the pseudo-inverse Jacobian is defined for systems that are not square or have full rank, and is a widely used solution to the inverse kinematics problem [7]-[9].

A robotic system is defined as kinematically redundant if it possesses more DOFs than those required to perform a certain task [10]. In this case, the “surplus” DOFs can be employed to achieve several tasks using Null-Space-Based (NSB) behavioral control, also known as multiple task-priority inverse kinematics [11]. This framework, which possesses nice stability qualities [12] and has been successfully implemented on several robotic systems [13], [14], has been developed for equality tasks, which specify exactly one desired value for given states of the system. However, for a general robotic system, several objectives may not be described as equality tasks, but as set-based tasks, which are tasks that have a desired set of values rather than one exact desired value (e.g. staying within joint limits or avoiding obstacles) [15]. As recognized in [16], the multiple task-priority inverse kinematics algorithm is not suitable to handle set-based tasks directly, and these tasks are therefore usually transformed into more restrictive equality constraints through potential fields or cost functions [17], [18].

A first attempt to systematically include set-based tasks in a prioritized task-regulation framework is proposed in [19] and further improved in [20]. To handle the set-based tasks, the algorithms in [19], [20] transform the inverse kinematics problem into a QP problem, and therefore they can not be utilized directly into the multiple task-priority inverse kinematics algorithm. In [21], set-based tasks are handled by resorting to proper activation and regularization functions, but has the limitation that set-based tasks are only considered with higher priority than the equality tasks of the system.

A method to include set-based tasks in the NSB framework is first introduced in [22], and further formalized and analyzed in [23]. The result is an on-line kinematic control algorithm that generates a real-time reference for the systems joint velocities. A set-based task is ignored while the task value is within its valid set, and the remaining tasks of the system then decide the trajectory. On the border of the valid set, the set-based task either remains ignored, or it is implemented as an equality task with the goal of freezing the task on the boundary. The proposed algorithm will choose the latter if the other tasks of the system push the set-based task out of its valid set. In the opposite case, the set-based task is still ignored. This results in a switched system with 2n modes, where n is the number of set-based tasks. The system is proven to achieve asymptotic convergence of all equality tasks and satisfaction of all high-priority set-based tasks, given that the tasks are linearly independent and the generated reference is followed [23]. The purpose of this paper is to further investigate and experimentally validate this algorithm and its properties proven in [23]. The set-based tasks that have been implemented are collision avoidance and field of view (FOV). Joint limit avoidance, limited workspace and manipulability are examples of other tasks suitable to be
considered as set-based tasks.

The experiments were run on a 6 DOF industrial manipulator from Universal Robots - the UR5. It is equipped with joint servo controllers produced by the robot manufacturer, and is both highly suitable and commonly used for experimental verification of control algorithms [24]-[27].

This paper is organized as follows: Section II describes the UR5, the overall control structure and communication protocol used in the experiments. Section III defines the tangent cone of a closed set, which is a crucial part of the implemented algorithm. The concrete implemented examples are presented in Section IV followed by the experimental results in Section V. Conclusions are given in Section VI.

In this paper, vectors and matrices are expressed in bold. Furthermore, a task is denoted as \( \sigma \) (or \( \Sigma \) for one-dimensional tasks). Equality tasks are marked with number subscripts and set-based with letters. Furthermore, \( \tilde{\sigma} = \sigma_{\text{des}} - \sigma \) denotes the task error, i.e. the difference between the desired and actual task value. \( J^*_a \) is the Moore-Penrose pseudoinverse of the Jacobian matrix \( J_a \), and \( N_a \equiv I - J^*_a J_a \) denotes the corresponding null-space matrix with the property \( J_a N_a = 0 \).

II. UR5 AND CONTROL SETUP

A. UR5 Kinematics

The UR5 is a manipulator with 6 revolute joints, and the joint angles are denoted \( \mathbf{q} \equiv [q_1, q_2, q_3, q_4, q_5, q_6]^T \). In this paper, the Denavit-Hartenberg (D-H) parameters are used to calculate the forward kinematics. The parameters are given in Table II and illustrated in Fig. 1.

<table>
<thead>
<tr>
<th>Joint</th>
<th>( a_i ) [m]</th>
<th>( d_i ) [rad]</th>
<th>( b_i ) [m]</th>
<th>( \theta_i ) [rad]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.089</td>
<td>0</td>
<td>( \pi/2 )</td>
</tr>
<tr>
<td>2</td>
<td>-0.425</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>-0.392</td>
<td>0</td>
<td>0</td>
<td>( \pi/2 )</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.109</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.095</td>
<td>0</td>
<td>( \pi/2 )</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.082</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TABLE I: Table of the D-H parameters of the UR5. The corresponding coordinate systems can be seen in Fig. 1.

![Fig. 1: Coordinate frames corresponding to the D-H parameters in Table I. Illustration from [24].](image1)

B. Control Setup and Communication Protocol

The UR5 is equipped with a high-level controller that can control the robot both in joint and Cartesian space. In the experiments presented here, a calculated reference \( \mathbf{q}_{\text{ref}} \) is sent to the high-level controller, which is assumed to function nominally such that

\[
\mathbf{q} \approx \mathbf{q}_{\text{ref}}.
\]

From this reference, \( \mathbf{q}_{\text{off}} \) and \( \mathbf{q}_{\text{on}} \) are extrapolated and sent with \( \mathbf{q}_{\text{ref}} \) to the low-level controller.

The structure of the system is illustrated in Fig. 2. The algorithm from [23] is implemented in the kinematic controller block. Every timestep, a reference for the joint velocities is calculated and integrated to desired joint angles \( \mathbf{q}_{\text{ref}} \). This is used as feedback to close the kinematic loop and as input to the dynamic controller, which in turn applies torques to the joint motors. The communication between the implemented algorithm and the industrial manipulator system occurs through a TCP/IP connection which operates at 125 Hz. The algorithm itself is implemented in Python.

![Fig. 2: The control structure of the experiments. The tested algorithm is implemented in the kinematic controller block.](image2)

III. TANGENT CONE

This section presents the tangent cone of a closed set, which is used in the implementation to decide which set-based tasks are activated. For a detailed description of the algorithm and theoretical background, the interested reader is referred to [23].

The tangent cone of a closed set \( C \)

\[
C = [a, b]
\]

is defined as

\[
T_C(\sigma) = \begin{cases} [0, \infty) & \sigma = a \\ \mathbb{R} & \sigma \in P \\ (-\infty, 0] & \sigma = b \end{cases}
\]

where \( P \) is the interior of \( C \).

In the implementation, the following boolean function is used to check if the time-derivative of a set-based task \( \sigma \) with a valid set \( C = [\sigma_{\text{min}}, \sigma_{\text{max}}] \) is in the tangent cone of \( C \), i.e. \( \tilde{\sigma} \in T_C(\sigma) \):

```python
bool in_T_C(sigma_dot, sigma, sigma_min, sigma_max):
    if sigma > sigma_min and sigma < sigma_max:
        return True
    else if sigma <= sigma_min and sigma_dot >= 0:
        return False
    else if sigma >= sigma_min and sigma_dot < 0:
        return False
    else if sigma <= sigma_max and sigma_dot <= 0:
        return False
    else if sigma >= sigma_max and sigma_dot > 0:
        return False
```

in_T_C is illustrated in Fig. 3.
IV. IMPLEMENTED EXAMPLES

In this section, three individual implemented tasks (position control, collision avoidance and field of view) are defined and numeric values are given for their desired value/valid set (Section IV-A). These tasks have then been implemented in three examples with different combinations of task priority and set-based/equality tasks. The examples are described in Sections IV-B to IV-D.

A. Implemented Tasks

Three tasks make up the basis for the experiments: Position control, collision avoidance, and FOV. In the examples, position control is always implemented as an equality task and collision avoidance as a set-based task. FOV has been implemented as both an equality (Example 1) and a set-based (Examples 2 and 3) task.

1) Position Control: The position of the end effector relative to the base coordinate frame is given by the forward kinematics. The analytical expression can be found through the homogeneous transformation matrix using the D-H parameters given in Table I

\[ \mathbf{\sigma}_{pos} = f(q) \in \mathbb{R}^3 \]

\[ \mathbf{\sigma}_{pos} = J_{pos}(q)q = \frac{df}{dq}q \]

In these experiments, the system has been given two waypoints for the end effector to reach:

\[ \mathbf{p}_{o1} = [0.486 \ m \ -0.066 \ m \ -0.250 \ m]^T \] and \( \mathbf{p}_{o2} = [0.320 \ m \ 0.370 \ m \ -0.250 \ m]^T \) (7)

A circle of acceptance (COA) of 0.02 m is implemented for switching from \( \mathbf{\sigma}_{pos,des} = \mathbf{p}_{o1} \) to \( \mathbf{\sigma}_{pos,des} = \mathbf{p}_{o2} \). The task gain matrix has been chosen as

\[ \Lambda_{pos} = \text{diag}(0.3, 0.3, 0.3). \]

This relatively low task gain it to ensure that the position task does not ask for too great joint velocities even when the task error is large, which would cause the UR5 to enter a security stop mode.

2) Collision Avoidance: To avoid a collision between the end effector and an object at position \( \mathbf{p}_o \in \mathbb{R}^3 \), the distance between them is used as a task:

\[ \mathbf{\sigma}_{CA} = \sqrt{(\mathbf{p}_o - \mathbf{\sigma}_{pos})^T(\mathbf{p}_o - \mathbf{\sigma}_{pos})} \in \mathbb{R} \] (9)

\[ \mathbf{\sigma}_{CA} = J_{CA}(q)q = -\frac{\mathbf{p}_o - \mathbf{\sigma}_{pos}}{\mathbf{\sigma}_{CA}} J_{pos}(q)q \] (10)

In all experiments, two obstacles have been introduced, hence two collision avoidance tasks are necessary. The obstacles are positioned at

\[ \mathbf{p}_{o1} = [0.40 \ m \ -0.25 \ m \ -0.33 \ m]^T \] and \( \mathbf{p}_{o2} = [0.40 \ m \ 0.15 \ m \ -0.33 \ m]^T \) (12)

and have a radius of 0.18 m and 0.15 m, respectively. This radius is used as the minimum value of the set-based collision avoidance task to ensure that the end effector is never closer to the obstacle center than the allowed radius, see Table II.

[IV] For the collision avoidance task, it is only necessary to ensure that the distance between the obstacle and end effector is greater than some radius to avoid collision, and hence this task does not have a maximum limit. Because the task is only considered as a high-priority set-based task, it is not necessary to choose a task gain.

3) Field of View: The field of view is defined as the outgoing vector of the end effector, i.e. the \( z_e \)-axis in Fig. 1. This vector expressed in base coordinates is denoted \( \mathbf{a} \in \mathbb{R}^3 \), and can be found through the homogeneous transformation matrix using the D-H parameters.

\[ \mathbf{a} = \mathbf{g}(q) \in \mathbb{R}^3 \] (13)

\[ \mathbf{a} = J_{FOV,3DOF}(q)q = \frac{d\mathbf{g}}{dq}q \] (14)

FOV is a useful task when directional devices or sensors are mounted on the end-effector and they are desired to point in a certain direction \( \mathbf{a}_{des} \in \mathbb{R}^3 \). In these experiments,

\[ \mathbf{a}_{des} \equiv \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \] (15)

is constant, and the task is defined as the norm of the error between \( \mathbf{a} \) and \( \mathbf{a}_{des} \):

\[ \mathbf{\sigma}_{FOV} = \sqrt{(\mathbf{a}_{des} - \mathbf{a})^T(\mathbf{a}_{des} - \mathbf{a})} \in \mathbb{R} \] (16)

\[ \mathbf{\sigma}_{FOV} = J_{FOV}(q)q = -\frac{(\mathbf{a}_{des} - \mathbf{a})^T}{\mathbf{\sigma}_{FOV}} J_{FOV,3DOF}(q)q \] (17)

In Example 1, FOV is implemented as an equality task. Since \( \mathbf{\sigma}_{FOV} \) is defined as the norm of the error, \( \mathbf{\sigma}_{FOV,des} = 0 \). In Examples 2 and 3, FOV is implemented as a set-based task with a maximum value to limit the error between \( \mathbf{a} \) and \( \mathbf{a}_{des} \). Here, the maximum value for the set-based FOV task is set as 0.2622. This corresponds to allowing the angle between \( \mathbf{a}_{des} \) and \( \mathbf{a} \) being 15° or less. The gain for this task is chosen as

\[ \Lambda_{FOV} = 1. \] (18)

Note in (10) and (17) that \( J_{CA} \) and \( J_{FOV} \) are not defined for \( \mathbf{\sigma}_{CA} = 0 \) and \( \mathbf{\sigma}_{FOV} = 0 \), respectively. In the implementation, this is solved by adding a small \( \varepsilon > 0 \) to the denominator of these two Jacobians thereby ensuring that division by zero does not occur.
B. Example 1

In Example 1, FOV is implemented as an equality task, and the system has two set-based tasks.

\[ \sigma \]

According to [23], a system with 2 set-based tasks have \( 2^2 = 4 \) modes to consider: One containing only the equality tasks, one where \( \sigma_a \) is frozen, one where \( \sigma_b \) is frozen and one where \( \sigma_a \) and \( \sigma_b \) are frozen. However, in this case, the two obstacles have no points of intersection. Hence, it will never be necessary to freeze both \( \sigma_a \) and \( \sigma_b \), and thus the system has three modes:

- **Mode 1:** \( q_{ref} = f_1 \triangleq J_1^f A_1 \sigma_1 \) \( + N_1 J_2^f A_2 \sigma_2 \) (19)
- **Mode 2:** \( q_{ref} = f_2 \triangleq N_2 J_1^f A_1 \sigma_1 \) \( + N_4 J_2^f A_2 \sigma_2 \) (20)
- **Mode 3:** \( q_{ref} = f_3 \triangleq N_2 J_1^f A_1 \sigma_1 \) \( + N_3 J_2^f A_2 \sigma_2 \) (21)

Using the fact that \( \sigma_1 = J_1 f_1 \) and \( \sigma_0 = J_0 f_1 \) in mode 1, the pseudocode below illustrates the implementation of the system:

```python
a = in_T_C(J_a*f1,sigma_a,0.18,infty)
b = in_T_C(J_b*f1,sigma_b,0.15,infty)

if a==True and b==True
    mode = 1
    q_dot_ref = f1
else if b==False
    mode = 2
    q_dot_ref = f2
else if a==False
    mode = 3
    q_dot_ref = f3
```

C. Example 2

In Example 2, FOV is implemented as a high-priority set-based task, and the system has three set-based tasks in total.

\[ \sigma \]

According to [23], this should result in \( 2^3 = 8 \) modes to consider. However, as in Example 1, we can discard the two modes where both \( \sigma_a \) and \( \sigma_b \) are frozen. Thus, 6 modes have to be considered:

<table>
<thead>
<tr>
<th>Name</th>
<th>Task description</th>
<th>Type</th>
<th>Valid set C</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_a )</td>
<td>Collision avoidance</td>
<td>Set-based</td>
<td>( C_a = (0.18, \infty) )</td>
</tr>
<tr>
<td>( \sigma_b )</td>
<td>Collision avoidance</td>
<td>Set-based</td>
<td>( C_b = (0.15, \infty) )</td>
</tr>
<tr>
<td>( \sigma_c )</td>
<td>Field of view</td>
<td>Equality</td>
<td>( C_c = (-\infty, 0.2622] )</td>
</tr>
<tr>
<td>( \sigma_1 )</td>
<td>Position</td>
<td>Set-based</td>
<td>-</td>
</tr>
</tbody>
</table>

**TABLE III:** Implemented tasks in Example 2 sorted by decreasing priority.

D. Example 3

In Example 3, FOV is implemented as a low-priority set-based task.

\[ \sigma \]

The implementation is very similar to Example 2. However, lower-priority set-based tasks can not be guaranteed to be satisfied at all times [23]. Hence, if the FOV error exceeds the maximum value of 0.2622, rather than attempting to freeze the task at its current value, an effort is made to push it back to the boundary of the valid set:

\[ \sigma_c = \sigma_{FOV, \text{max}} - \sigma_c = 0.2622 - \sigma_c \] (28)
End effector trajectory. Obstacles shown as spheres and waypoints as red dots.

Active mode over time.

Distance to obstacle centers over time.

Norm of the error between desired and actual FOV vectors over time.

Fig. 4: Logged data from experimental results.

Mode 1: \[ \dot{q}_{\text{ref}} = f_1 \triangleq J_1^T \Lambda_1 \dot{\sigma}_1 \]

Mode 2: \[ \dot{q}_{\text{ref}} = f_2 \triangleq N_1 J_2^T \Lambda_1 \dot{\sigma}_1 \]

Mode 3: \[ \dot{q}_{\text{ref}} = f_3 \triangleq N_0 J_0^T \Lambda_1 \dot{\sigma}_1 \]

Mode 4: \[ \dot{q}_{\text{ref}} = f_4 \triangleq J_4^T \Lambda_1 \dot{\sigma}_1 + N_1 J_4^T \Lambda_c \dot{\sigma}_c \]

Mode 5: \[ \dot{q}_{\text{ref}} = f_5 \triangleq N_1 J_5^T \Lambda_1 \dot{\sigma}_1 + N_1 J_5^T \Lambda_c \dot{\sigma}_c \]

Mode 6: \[ \dot{q}_{\text{ref}} = f_6 \triangleq N_0 J_0^T \Lambda_1 \dot{\sigma}_1 + N_0 J_0^T \Lambda_c \dot{\sigma}_c \]

Similarly to Example 2, \( \dot{\sigma}_1 \) and \( \dot{\sigma}_c \) are frozen in modes 2 and 5, 3 and 6 respectively. However, since \( \dot{\sigma}_1 \) is a low-priority set-based task, it can not be guaranteed that it is frozen on the border of \( C_c \). Therefore, unlike Example 2, \( \text{in}_T \mathcal{C} \) might return \texttt{False} in modes 4, 5 and 6 with \( \sigma_c \) as input. Even so, in these modes, an attempt is made to satisfy the set-based task by actively pushing \( \sigma_c \) back to the border of \( C_c \) and even if this attempt is unsuccessful \( \text{in}_T \mathcal{C} = \text{False} \) with \( \sigma_c \) as input), it is not due to the task not being handled, but because it is a lower-priority task. Therefore, this condition is not checked in modes 4, 5 and 6, and the implementation is identical to Example 2 with modes defined by (29)-(34).
Fig. 5: Pictures from simulation and actual experiments, Example 1. In the simulation, the base and end effector coordinate system is illustrated with green, blue and red axes for the x-, y- and z-axes respectively. These correspond to the coordinate frames of the actual robot.

V. EXPERIMENTAL RESULTS

This section presents the results from running Examples 1-3 on the UR5 manipulator. The relevant logged data is illustrated in Fig. 4, and Fig. 5 displays screenshots/images from simulation and actual experiment from Example 1.

In all examples, the position task is fulfilled as predicted by the theory [23], i.e. the two waypoints are reached by the end effector. Furthermore, the end effector avoids the two obstacles by locking the distance to the obstacle center at the obstacle radius until the other active tasks drive the end effector away from the obstacle center on their own accord. This can be seen in Figures 4a-4c, and is also confirmed by Figures 4g-4i: The set-based collision avoidance tasks never exceed the valid sets $C_a$ and $C_b$, but freeze on the boundary of these sets.

Figures 4d-4f display the active mode over time, and confirm that mode changes coincide with set-based tasks either being activated (frozen on boundary/leaving valid set) or deactivated (unfrozen/approaching valid set). An increase in mode means a new set-based task has been activated and vice versa.

In Example 1, FOV is implemented as an equality task with lower priority than the position task with the goal of aligning the FOV vector $a$ with $a_{\text{des}} = [1 \ 0 \ 0]^T$. This corresponds to the z-axis of the end effector being parallel to the x-axis of the base coordinate system. As can be seen in Fig. 4h, the norm of the error between $a$ and $a_{\text{des}}$ converges to zero at about $t = 30$ s, and Fig. 5 shows that in the end configuration, these two vectors are indeed parallel.

In Example 2, FOV is a high-priority set-based task with the same maximum value as Example 2. By comparing Figures 4k and 4l it is evident that these implementations behave similarly until $t = 13$ s, when the system enters mode 4 and $a_{\text{c}}$ is frozen because the error between the actual and desired FOV vectors has reached its upper limit and keeping the task deactivated would result in the maximum value being violated. Shortly after, the end effector reaches the second obstacle, and so mode 6 is activated where both $a_{\text{a}}$ and $a_{\text{c}}$ are frozen. Once the end effector has moved around the obstacle, $a_{\text{a}}$ is released. $a_{\text{c}}$, however, can not be released without leaving $C_c$, and so the system goes back to mode 4 and remains there for the duration of the example.

In Example 3, FOV is a low-priority set-based task with the same maximum value as Example 2. By comparing Figures 4k and 4l it is evident that these implementations behave similarly until $t = 13$ s, when the system enters mode 4 and activates $a_{\text{c}}$. In Example 2, the task is frozen on the boundary, which is guaranteed due to the fact it is high priority. As explained in Section IV, low priority set-based tasks can not be guaranteed to actually freeze on the boundary, and they are therefore activated with the goal of pushing the task back to its boundary. This is confirmed by Fig. 4l. In this example, $a_{\text{c}}$ does indeed exceed its maximum value in spite of the system activating the task. However, eventually $a_{\text{c}}$ converges back to the boundary of $C_c$.

Figures 4j and 4l show that implementing FOV as a lower priority equality and set-based render similar results. As expected, the equality task converges to the exact desired value and the set-based to the boundary of the valid set. Even so, in the case that the system has several other tasks with even lower priority, it might be beneficial to implement
FOV as a set-based task as this imposes less constraint on the lower-priority tasks when inactive.

VI. CONCLUSIONS
A method proposed in [22], [23] for incorporating set-based tasks in the singularity-robust multiple task-priority inverse kinematics framework has been illustrated and validated in this paper. In particular, the method has been implemented on a 6 DOF UR5 manipulator. Three examples have been constructed to test various qualities and the performance of the algorithm. In summary, the experimental results confirm the following properties:

- All equality tasks converge to their desired value.
- All high-priority set-based tasks with initial conditions in their valid set C stay in this set ∀t ≥ 0.
- All high-priority set-based tasks with initial conditions outside C can only 1) freeze at the current value, or 2) move closer to C. Hence, the initial condition is the maximum deviation from C.
- If a high-priority set-based task with initial conditions outside its valid set eventually enters C, it will stay in this set ∀t ≥ t_e, where t_e is the time the task entered the set.
- Lower-priority set-based tasks are not necessarily satisfied.

Furthermore, it is suggested that implementing a low-priority task as set-based rather than as an equality task is less restrictive on even lower priority tasks as they are not affected by the set-based task when it is not active. This remains a topic for future work.

ACKNOWLEDGMENTS
This work was supported by the Research Council of Norway through the Center of Excellence funding scheme, project number 223254, and by the European Community through the projects ARCAS (FP7-287617), EuRoC (FP7-608849), DexROV (H2020-635491) and AEROARMS (H2020-644271). The authors would also like to thank Magnus C. Bjerken at Sintef ICT, Department of Applied Cybernetics, for sharing advice and experience regarding implementation on the UR5 manipulator.

REFERENCES