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

Signe Moe[1], Gianluca Antonelli[2], Kristin Y. Pettersen[1] and Johannes Schrimpf[3]

***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 2^n 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

[1]S.Moe and K.Y.Pettersen are with the Center for Autonomous Marine Operations and Systems (AMOS), at The Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway {signe.moe, kristin.y.pettersen}@itk.ntnu.no

[2]G.Antonelli is with the Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, Cassino, Italy antonelli@unicas.it

[3]J.Schrimpf is with The Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway johannes.schrimpf@itk.ntnu.no

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 $\boldsymbol{\sigma}$ (or σ for one-dimensional tasks). Equality tasks are marked with number subscripts and set-based with letters. Furthermore, $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_{\text{des}} - \boldsymbol{\sigma}$ denotes the task error, i.e. the difference between the desired and actual task value. $\boldsymbol{J}_{\text{a}}^{\dagger}$ is the MoorePenrose pseudoinverse of the Jacobian matrix $\boldsymbol{J}_{\text{a}}$, and $\boldsymbol{N}_{\text{a}} \triangleq \boldsymbol{I} - \boldsymbol{J}_{\text{a}}^{\dagger}\boldsymbol{J}_{\text{a}}$ denotes the corresponding null-space matrix with the property $\boldsymbol{J}_{\text{a}}\boldsymbol{N}_{\text{a}} \equiv \mathbf{0}$.

II. UR5 AND CONTROL SETUP

A. UR5 Kinematics

The UR5 is a manipulator with 6 revolute joints, and the joint angles are denoted $\boldsymbol{q} \triangleq \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^{\text{T}}$. In this paper, the Denavit-Hartenberg (D-H) parameters are used to calculate the forward kinematics. The parameters are given in Table I and illustrated in Fig. 1.

Joint	a_i [m]	α_i [rad]	d_i [m]	θ_i [rad]
1	0	$\pi/2$	0.089	q_1
2	-0.425	0	0	q_2
3	-0.392	0	0	q_3
4	0	$\pi/2$	0.109	q_4
5	0	$-\pi/2$	0.095	q_5
6	0	0	0.082	q_6

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

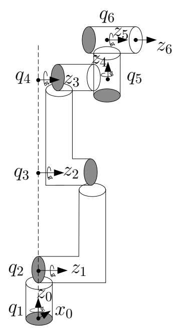


Fig. 1: Coordinate frames corresponding to the D-H parameters in Table I. Illustration from [24].

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 $\boldsymbol{q}_{\text{ref}}$ is sent to the high-level controller, which is assumed to function nominally such that

$$\boldsymbol{q} \approx \boldsymbol{q}_{\text{ref}}. \tag{1}$$

From this reference, $\dot{\boldsymbol{q}}_{\text{ref}}$ and $\ddot{\boldsymbol{q}}_{\text{ref}}$ are extrapolated and sent with $\boldsymbol{q}_{\text{ref}}$ 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 $\boldsymbol{q}_{\text{ref}}$. 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.

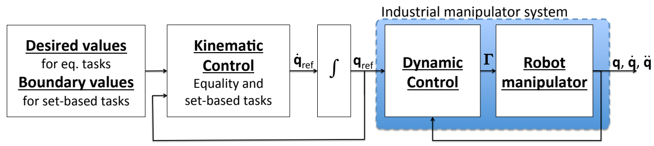


Fig. 2: The control structure of the experiments. The tested algorithm is implemented in the kinematic controller block.

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

$$C = [a, b] \tag{2}$$

is defined as

$$T_C(\sigma) = \begin{cases} [0, \infty) & \sigma = a \\ \mathbb{R} & \sigma \in P \\ (-\infty, 0] & \sigma = b \end{cases} \tag{3}$$

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 σ with a valid set $C = [\sigma_{\min}, \sigma_{\max}]$ is in the tangent cone of C, i.e. $\dot{\sigma} \in T_C(\sigma)$:

```
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 True
   else if sigma <= sigma_min and sigma_dot < 0
      return False
   else if sigma >= sigma_max and sigma_dot <= 0
      return True
   else if sigma >= sigma_max and sigma_dot > 0
      return False
```

`in_T_C` is illustrated in Fig. 3.

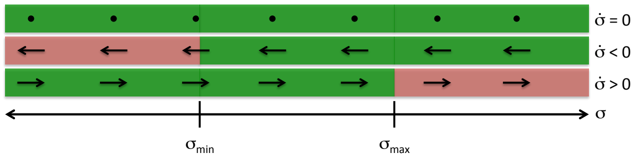


Fig. 3: Graphic illustration of `in_T_C` with return value `True` shown in green and `False` in red. The function only returns `False` when σ is outside or on the border of its valid set and the derivative points away from the valid set.

Note that in the implementation of the tangent cone, the border is not implemented as a perfect equality as in the definition (3), but as an inequality (i.e. $\sigma \leq \sigma_{\text{min}}$ rather than $\sigma = \sigma_{\text{min}}$). This is to handle small numerical inaccuracies that result from discretization of a the continuous system. Furthermore, given initial conditions outside the valid set C, the chosen implementation is still well-founded.

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.

$$\boldsymbol{\sigma}_{\text{pos}} = \boldsymbol{f}(\boldsymbol{q}) \in \mathbb{R}^3 \tag{4}$$

$$\dot{\boldsymbol{\sigma}}_{\text{pos}} = \boldsymbol{J}_{\text{pos}}(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{q}}\dot{\boldsymbol{q}} \tag{5}$$

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

$$\boldsymbol{p}_{\text{w1}} = \begin{bmatrix}0.486 \text{ m} & -0.066 \text{ m} & -0.250 \text{ m}\end{bmatrix}^{\text{T}}, \text{ and} \tag{6}$$

$$\boldsymbol{p}_{\text{w2}} = \begin{bmatrix}0.320 \text{ m} & 0.370 \text{ m} & -0.250 \text{ m}\end{bmatrix}^{\text{T}}. \tag{7}$$

A circle of acceptance (COA) of 0.02 m is implemented for switching from $\boldsymbol{\sigma}_{\text{pos,des}} = \boldsymbol{p}_{w1}$ to $\boldsymbol{\sigma}_{\text{pos,des}} = \boldsymbol{p}_{w2}$. The task gain matrix has been chosen as

$$\boldsymbol{\Lambda}_{\text{pos}} = \text{diag}(\begin{bmatrix}0.3 & 0.3 & 0.3\end{bmatrix}). \tag{8}$$

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 $\boldsymbol{p}_o \in \mathbb{R}^3$, the distance between them is used as a task:

$$\sigma_{\text{CA}} = \sqrt{(\boldsymbol{p}_o - \boldsymbol{\sigma}_{\text{pos}})^{\text{T}}(\boldsymbol{p}_o - \boldsymbol{\sigma}_{\text{pos}})} \in \mathbb{R} \tag{9}$$

$$\dot{\sigma}_{\text{CA}} = \boldsymbol{J}_{\text{CA}}(\boldsymbol{q})\dot{\boldsymbol{q}} = -\frac{(\boldsymbol{p}_o - \boldsymbol{\sigma}_{\text{pos}})^{\text{T}}}{\sigma_{\text{CA}}}\boldsymbol{J}_{\text{pos}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{10}$$

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

$$\boldsymbol{p}_{\text{o1}} = \begin{bmatrix}0.40 \text{ m} & -0.25 \text{ m} & -0.33 \text{ m}\end{bmatrix}^{\text{T}} \text{ and} \tag{11}$$

$$\boldsymbol{p}_{\text{o2}} = \begin{bmatrix}0.40 \text{ m} & 0.15 \text{ m} & -0.33 \text{ m}\end{bmatrix}^{\text{T}}, \tag{12}$$

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_6-axis in Fig. 1. This vector expressed in base coordinates is denoted $\boldsymbol{a} \in \mathbb{R}^3$, and can be found through the homogeneous transformation matrix using the D-H parameters.

$$\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^3 \tag{13}$$

$$\dot{\boldsymbol{a}} = \boldsymbol{J}_{\text{FOV,3DOF}}(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{g}}{\mathrm{d}\boldsymbol{q}}\dot{\boldsymbol{q}} \tag{14}$$

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 $\boldsymbol{a}_{\text{des}} \in \mathbb{R}^3$. In these experiments,

$$\boldsymbol{a}_{\text{des}} \equiv \begin{bmatrix}1 & 0 & 0\end{bmatrix}^{\text{T}} \tag{15}$$

is constant, and the task is defined as the norm of the error between $\boldsymbol{a}$ and $\boldsymbol{a}_{\text{des}}$:

$$\sigma_{\text{FOV}} = \sqrt{(\boldsymbol{a}_{\text{des}} - \boldsymbol{a})^{\text{T}}(\boldsymbol{a}_{\text{des}} - \boldsymbol{a})} \in \mathbb{R} \tag{16}$$

$$\dot{\sigma}_{\text{FOV}} = \boldsymbol{J}_{\text{FOV}}(\boldsymbol{q})\dot{\boldsymbol{q}} = -\frac{(\boldsymbol{a}_{\text{des}} - \boldsymbol{a})^{\text{T}}}{\sigma_{\text{FOV}}}\boldsymbol{J}_{\text{FOV,3DOF}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{17}$$

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

$$\Lambda_{\text{FOV}} = 1. \tag{18}$$

Note in (10) and (17) that $\boldsymbol{J}_{\text{CA}}$ and $\boldsymbol{J}_{\text{FOV}}$ are not defined for $\sigma_{\text{CA}} = 0$ and $\sigma_{\text{FOV}} = 0$, 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.

Name	Task description	Type	Valid set C
σ_{a}	Collision avoidance	Set-based	$C_{\mathrm{a}} = [0.18, \infty)$
σ_{b}	Collision avoidance	Set-based	$C_{\mathrm{b}} = [0.15, \infty)$
$\boldsymbol{\sigma}_1$	Position	Equality	-
σ_2	Field of view	Equality	-

TABLE II: Implemented tasks in Example 1 sorted by decreasing priority.

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 σ_{a} is frozen, one where σ_{b} is frozen and one where σ_{a} and σ_{b} are frozen. However, in this case, the two obstacles have no points of intersection. Hence, it will never be necessary to freeze both σ_{a} and σ_{b}, and thus the system has three modes:

$$\text{Mode 1:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_1 \triangleq \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_2^{\dagger}\Lambda_2\tilde{\sigma}_2 \tag{19}$$

$$\text{Mode 2:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_2 \triangleq \boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\Lambda_2\tilde{\sigma}_2 \tag{20}$$

$$\text{Mode 3:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_3 \triangleq \boldsymbol{N}_{\mathrm{b}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{b}1}\boldsymbol{J}_2^{\dagger}\Lambda_2\tilde{\sigma}_2 \tag{21}$$

Using the fact that $\dot{\sigma}_{\mathrm{a}} = \boldsymbol{J}_{\mathrm{a}}\boldsymbol{f}_1$ and $\dot{\sigma}_{\mathrm{b}} = \boldsymbol{J}_{\mathrm{b}}\boldsymbol{f}_1$ in mode 1, the pseudocode below illustrates the implementation of the system:

```
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 a==False
   mode = 2
   q_dot_ref = f2
else if b==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.

Name	Task description	Type	Valid set C
σ_{a}	Collision avoidance	Set-based	$C_{\mathrm{a}} = [0.18, \infty)$
σ_{b}	Collision avoidance	Set-based	$C_{\mathrm{b}} = [0.15, \infty)$
σ_{c}	Field of view	Set-based	$C_{\mathrm{c}} = (-\infty, 0.2622]$
$\boldsymbol{\sigma}_1$	Position	Equality	-

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

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 σ_{a} and σ_{b} are frozen. Thus, 6 modes have to be considered:

$$\text{Mode 1:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_1 \triangleq \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{22}$$

$$\text{Mode 2:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_2 \triangleq \boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{23}$$

$$\text{Mode 3:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_3 \triangleq \boldsymbol{N}_{\mathrm{b}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{24}$$

$$\text{Mode 4:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_4 \triangleq \boldsymbol{N}_{\mathrm{c}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{25}$$

$$\text{Mode 5:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_5 \triangleq \boldsymbol{N}_{\mathrm{ac}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{26}$$

$$\text{Mode 6:} \quad \dot{\boldsymbol{q}}_{\mathrm{ref}} = \boldsymbol{f}_6 \triangleq \boldsymbol{N}_{\mathrm{bc}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{27}$$

In mode 2 and 5 σ_{a} is frozen, meaning $\dot{\sigma}_{\mathrm{a}} \equiv 0$. Hence, in mode 2, `in_T_C` will always return `True` with σ_{a} as input, and it is therefore unnecessary to check this condition. The same applies to σ_{b} in modes 3 and 6, and σ_{c} in modes 4, 5 and 6. Thus, the pseudocode of the implementation is given below:

```
a1 = in_T_C(J_a*f1,sigma_a,0.18,infty)
b1 = in_T_C(J_b*f1,sigma_b,0.15,infty)
c1 = in_T_C(J_c*f1,sigma_c,-infty,0.2622)
b2 = in_T_C(J_b*f2,sigma_b,0.15,infty)
c2 = in_T_C(J_c*f2,sigma_c,-infty,0.2622)
a3 = in_T_C(J_a*f3,sigma_a,0.18,infty)
c3 = in_T_C(J_c*f3,sigma_c,-infty,0.2622)
a4 = in_T_C(J_a*f4,sigma_a,0.18,infty)
b4 = in_T_C(J_b*f4,sigma_b,0.15,infty)
b5 = in_T_C(J_b*f5,sigma_b,0.15,infty)
a6 = in_T_C(J_a*f6,sigma_a,0.18,infty)

if a1==True and b1==True and c1==True
   mode = 1
   q_dot_ref = f1
else if b2==True and c2==True
   mode = 2
   q_dot_ref = f2
else if a3==True and c3==True
   mode = 3
   q_dot_ref = f3
else if a4==True and b4==True
   mode = 4
   q_dot_ref = f4
else if b5==True
   mode = 5
   q_dot_ref = f5
else if a6==True
   mode = 6
   q_dot_ref = f6
```

D. Example 3

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

Name	Task description	Type	Valid set C
σ_{a}	Collision avoidance	Set-based	$C_{\mathrm{a}} = [0.18, \infty)$
σ_{b}	Collision avoidance	Set-based	$C_{\mathrm{b}} = [0.15, \infty)$
$\boldsymbol{\sigma}_1$	Position	Equality	-
σ_{c}	Field of view	Set-based	$C_{\mathrm{c}} = (-\infty, 0.2622]$

TABLE IV: Implemented tasks in Example 3 sorted by decreasing priority.

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:

$$\tilde{\sigma}_{\mathrm{c}} = \sigma_{\mathrm{FOV,max}} - \sigma_{\mathrm{c}} = 0.2622 - \sigma_{\mathrm{c}} \tag{28}$$

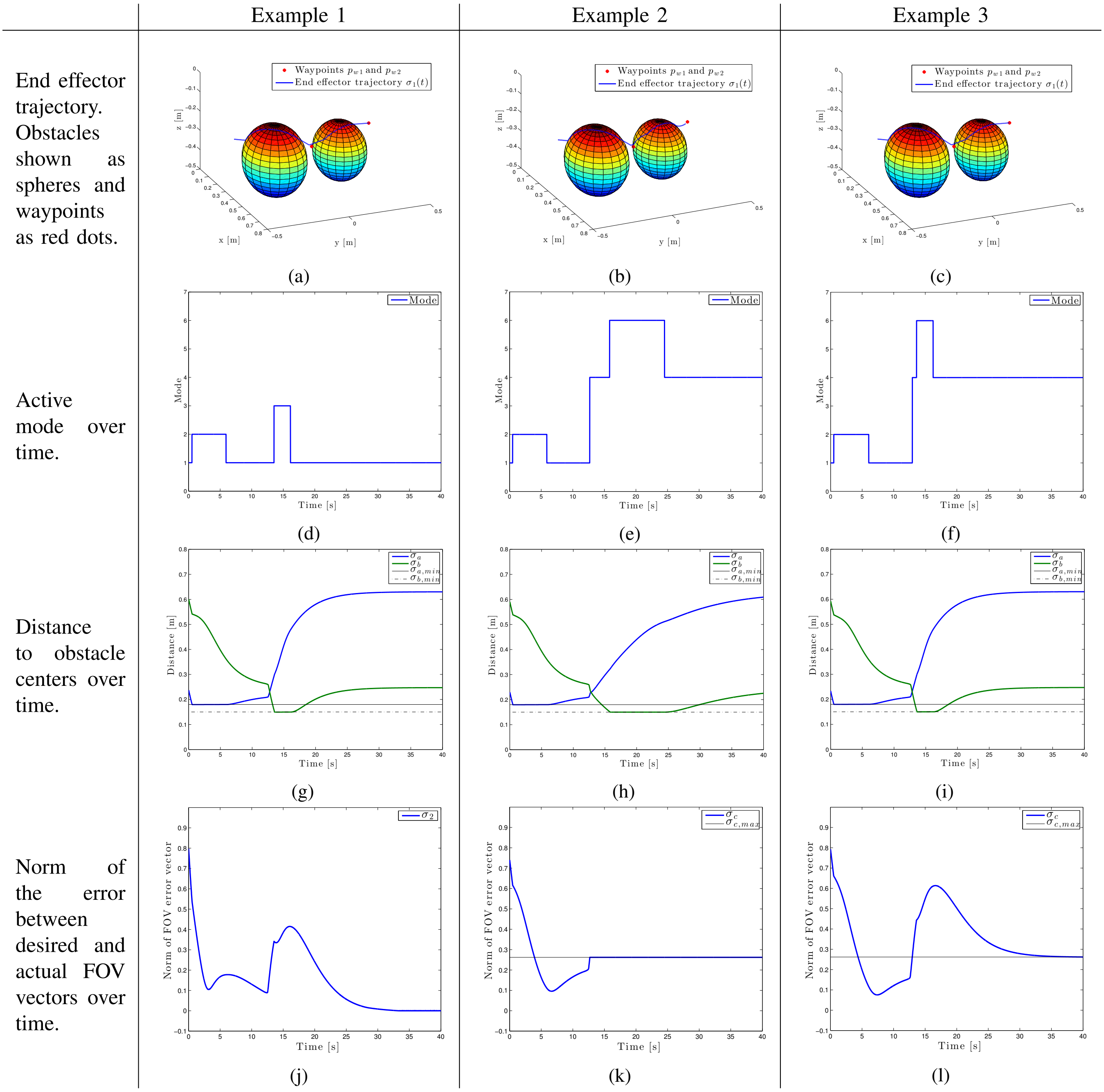


Fig. 4: Logged data from experimental results.

$$\text{Mode 1:} \quad \dot{\boldsymbol{q}}_{\text{ref}} = \boldsymbol{f}_1 \triangleq \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{29}$$

$$\text{Mode 2:} \quad \dot{\boldsymbol{q}}_{\text{ref}} = \boldsymbol{f}_2 \triangleq \boldsymbol{N}_{\text{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{30}$$

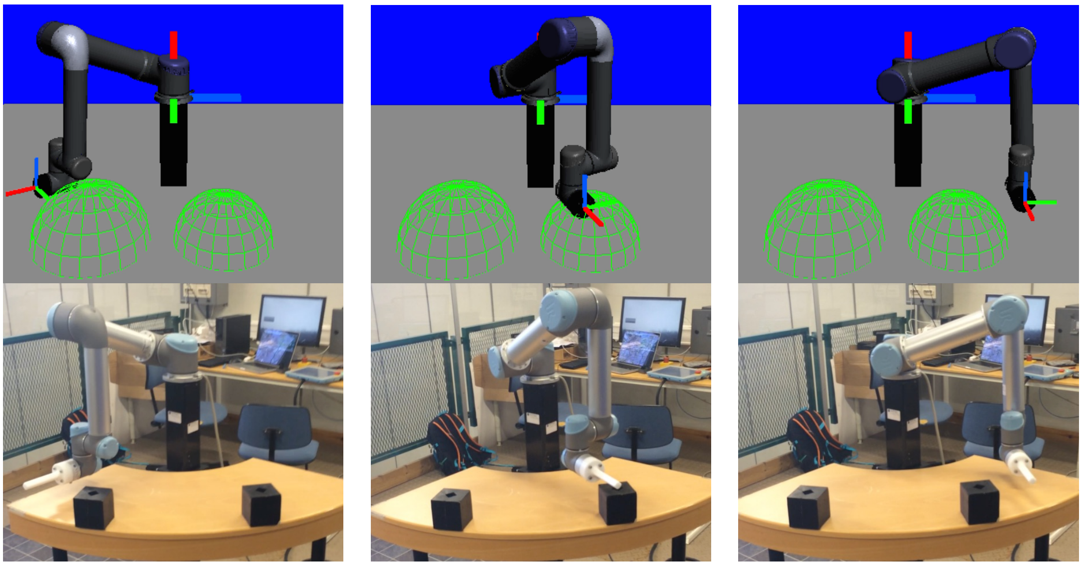
$$\text{Mode 3:} \quad \dot{\boldsymbol{q}}_{\text{ref}} = \boldsymbol{f}_3 \triangleq \boldsymbol{N}_{\text{b}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{31}$$

$$\text{Mode 4:} \quad \dot{\boldsymbol{q}}_{\text{ref}} = \boldsymbol{f}_4 \triangleq \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_{\text{c}}^{\dagger}\Lambda_{\text{c}}\tilde{\sigma}_{\text{c}} \tag{32}$$

$$\text{Mode 5:} \quad \dot{\boldsymbol{q}}_{\text{ref}} = \boldsymbol{f}_5 \triangleq \boldsymbol{N}_{\text{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\text{a}1}\boldsymbol{J}_{\text{c}}^{\dagger}\Lambda_{\text{c}}\tilde{\sigma}_{\text{c}} \tag{33}$$

$$\text{Mode 6:} \quad \dot{\boldsymbol{q}}_{\text{ref}} = \boldsymbol{f}_6 \triangleq \boldsymbol{N}_{\text{b}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\text{b}1}\boldsymbol{J}_{\text{c}}^{\dagger}\Lambda_{\text{c}}\tilde{\sigma}_{\text{c}} \tag{34}$$

Similarly to Example 2, σ_{a} and σ_{b} are frozen in modes 2 and 5, 3 and 6 respectively. However, since σ_{c} 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, `in_T_C` might return `False` in modes 4, 5 and 6 with σ_{c} as input. Even so, in these modes, an attempt is made to satisfy the set-based task by actively pushing σ_{c} back to the border of C_{c} and even if this attempt is unsuccessful (`in_T_C = False` with σ_{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).



(a) $t = 0$ s (b) $t = 16$ s (c) $t = 40$ s

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 $\boldsymbol{a}$ with $\boldsymbol{a}_{des} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$. 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. 4j, the norm of the error between $\boldsymbol{a}$ and $\boldsymbol{a}_{des}$ 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 a maximum value of 0.2622, corresponding to the angle between $\boldsymbol{a}$ and $\boldsymbol{a}_{des}$ not exceeding 15°. The task has initial conditions outside its valid set C_c (Fig. 4k). However, the other active tasks naturally bring the FOV closer to and eventually (at around $t = 4$ s) into C_c, and thus it is not necessary to freeze σ_c. Once σ_c enters C_c, the task will always stay in this set. At around $t = 13$ s, the system enters mode 4 and σ_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 σ_b and σ_c are frozen. Once the end effector has moved around the obstacle, σ_b is released. σ_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 σ_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, σ_c does indeed exceed its maximum value in spite of the system activating the task. However, eventually σ_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 $\forall t \geq 0$.
- 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 $\forall t \geq t_e$, 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. Bjerkeng at Sintef ICT, Department of Applied Cybernetics, for sharing advice and experience regarding implementation on the UR5 manipulator.

REFERENCES

[1] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[2] G. Antonelli, *Underwater Robots, Third Edition*. Springer International Publishing, 2014.

[3] K. Valavanis, P. Oh, and L. A. Piegla, *Unmanned Aircraft Systems*. Springer Netherlands, 2009.

[4] D. Nenchev, Y. Umetani, and K. Yoshida, "Analysis of a redundant free-flying spacecraft/manipulator system," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 1–6, 1992.

[5] F. Caccavale and B. Siciliano, "Kinematic control of redundant free-floating robotic systems," *Advanced Robotics*, vol. 15, no. 4, pp. 429–448, 2001.

[6] O. Egeland and K. Pettersen, "Free-floating robotic systems," in *Control Problems in Robotics and Automation*. Springer Berlin Heidelberg, 1998, vol. 230, pp. 119–134.

[7] C. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 2, pp. 245–250, 1983.

[8] S. Chiaverini, G. Oriolo, and I. D. Walker, *Springer Handbook of Robotics*. Heidelberg, D: B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, 2008, ch. Kinematically Redundant Manipulators, pp. 245–268.

[9] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, pp. 1–19, 2004.

[10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Verlag, 2009.

[11] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The Null-Space-Based behavioral control for autonomous robotic systems," *Intelligent Service Robotics*, vol. 1, no. 1, pp. 27–39, 2008.

[12] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 985–994, October, 2009.

[13] F. Arrichiello, J. Das, H. Heidarsson, S. Chiaverini, and G. Sukhatme, "Experiments in autonomous navigation with an under-actuated surface vessel via the null-space based behavioral control," in *Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2009)*, July 2009, pp. 362–367.

[14] F. Arrichiello, S. Chiaverini, G. Indiveri, and P. Pedone, "The null-space based behavioral control for a team of cooperative mobile robots with actuator saturations," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, Oct 2009, pp. 5911–5916.

[15] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," in *Proc. 8th ZFAC World Congress*, 1981.

[16] O. Kanoun, F. Lamiraux, and P. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[17] O. Khatib, "The potential field approach and operational space formulation in robot control," in *Adaptive and Learning Systems: Theory and Applications*, K. S. Narendra, Ed. Springer US, 1986.

[18] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, 1987, pp. 1152–1159.

[19] O. Kanoun, F. Lamiraux, and P. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *Robotics, IEEE Transactions on*, vol. 27, no. 4, pp. 785–792, 2011.

[20] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *International Journal of Robotics Research*, 2013.

[21] E. Simetti, G. Casalino, S. Torelli, A. Sperindé, and A. Turetta, "Floating Underwater Manipulation: Developed Control Methodology and Experimental Validation within the TRIDENT Project," *Journal of Field Robotics*, vol. 31, no. 3, pp. 364–385, 2013.

[22] G. Antonelli, S. Moe, and K. Pettersen, "Incorporating set-based control within the singularity-robust multiple task-priority inverse kinematics," in *Proc. 23rd Mediterranean Conference on Control and Automation*, Torremolinos, Spain, 2015.

[23] S. Moe, A. Teel, G. Antonelli, and K. Pettersen, "Stability analysis for set-based control within the singularity-robust multiple task-priority inverse kinematics framework," in *Proc. 54th IEEE Conference on Decision and Control*, 2015.

[24] H. Wu, W. Tizzano, T. T. Andersen, N. A. Andersen, and O. Ravn, "Hand-eye calibration and inverse kinematics of robot arm using neural network," *Advances in Intelligent Systems and Computing*, vol. 274, pp. 581–591, 2014.

[25] J. J. H. Lee, K. Frey, R. Fitch, and S. Sukkarieh, "Fast path planning for precision weeding," in *Proc. of Australasian Conference on Robotics and Automation*, Melbourne, Australia, 2014.

[26] S. Moe and I. Schjolberg, "Real-time hand guiding of industrial manipulator in 5 dof using microsoft kinect and accelerometer," in *Proc. 22th International Symposium on Robot and Human Interactive Communication (IEEE RO-MAN)*, Gyeongju, Korea, 2013, pp. 644–649.

[27] M. Bjerkeng, J. Schrimpf, T. Myhre, and K. Pettersen, "Fast dual-arm manipulation using variable admittance control: Implementation and experimental results," in *Proc. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, Sept 2014, pp. 4728–4734.