Object Recognition by Physical Properties Detection using Fault Reconstruction Techniques

Marco Negrete and Jesús Savage

RCF-MathWorks Fund - Call 2020
Technical Report

1 Introduction

1.1 Motivation

Consider the following serving-drinks-test scenario in the Robocup@Home league: a robot is asked to bring beverages and to do so, the robot performs the following common steps: it navigates to the bar, recognizes the beverages (usually cans or tetra packs), calculate some sort of inverse kinematics, moves its manipulator, grasps the object and returns to the user's location. This is a common situation in @Home tasks, nevertheless, what happens if there are empty cans or tetra packs in the bar? As it is commonly implemented in @Home league, robots are unable to distinguish between empty and full cans, since they commonly rely only in visual information of the beverage containers (see, for example, description papers of last edition winner teams [10, 7, 12]).

1.2 Objectives and goals

The project we propose consist in applying model based fault reconstruction techniques [3] to estimate the weight of the object being carried by the robot. If we consider the manipulator without load in the final effector as the nominal system, and the weight of the object as an external perturbation causing a faulty behavior, we can apply fault reconstruction techniques to estimate such perturbation, i.e., to estimate object's weight. There are several approaches for model-based fault reconstruction. Residual generation is a common technique where an observer is designed to be sensitive to fault signals. On the other hand, Sliding Mode Observers (SMO) are dynamic estimators designed to be robust against fault signals [9]. SMOs also provide the ability to reconstruct the fault signal by filtering the so called output error injection term [1]. We the estimation of the manipulated object mass, we expect to improve the manipulation performance in domestic service robots.

1.3 Description of the document

This technical report is organized as follows: In section 2 we briefly introduce the concept of Sliding Mode Observers (SMO) and we describe the robot Justina's manipulator, the hardware whose features we chose to test our proposal. In

section 3 we explain the theoretical part of our proposal while section 4 is dedicated to describe the implementation using several Simulink Toolboxes. Section 5 describe our results and also, the cases where the proposal can fail. In section 6 we show the reusability of our proposal by replicating some results using a simulated Katana manipulator. Finally in section 7 we state our conclusions and sketch the future work.

2 Background

2.1 Sliding Mode Observers

Sliding mode observers are disticontinuos observers that have the properties of finite time convergence and accurate tracking of the measured states once the sliding surface is reached. These kind of observers can be used to reconstruct faults or disturbance signals through an appropiate filtering of the so-called injection output error. Interested readers can find further information about Sliding Mode Observers in [9, 2, 4].

Fault detection and isolation, from the control theory approach, is commonly achieved by the implementation of residual generators. A common way to design such residual generators is through observers that are sensitive to the fault signals. Nevertheless, SMOs inherit the properties of Sliding Modes theory, i.e., SMOs are designed to be robust against disturbances and fault signals and to accurately track the system states. As previously explained, the discontinuos term of the SMO will contain information about the fault signal wich can be reconstructed by obtaining the equivalent output error injection.

2.2 Robot Justina's manipulator

Justina is a domestic service robot built at the Biorobotics Laboratory of the National Autonomous University of Mexico and developed under the ViRBot architecture [8]. This robot and its predecessors have been participating in the Robocup@Home league [11] since 2006 performing several tasks such as cleaning a table, serving drinks and several other tasks that humans ask for.

Among other actuators, Justina has two 7-DOF manipulators built with Dynamixel servomotors. Such servomotors allow the user to control them in several forms: by setting a goal position, setting a goal movement speed or setting a desired current which will produce a torque. This latter option is used in this project. Also, these motors provide position sensing with a resolution of less than a tenth of degree. Figure 1a shows robot Justina and its sensors and actuators.

To describe the kinematic chain and some physical properties of the manipulator, we use an URDF file. We already had an URDF describing the whole robot kinematic chain, nevertheless, for this project it was necessary only one of the mainpulators. We chose the left arm and extracted the corresponding tags to a new file. This file was used as a starting point for modeling and simulation purposes. Figure 1b shows the left arm of robot Justina as represented in the URDF.
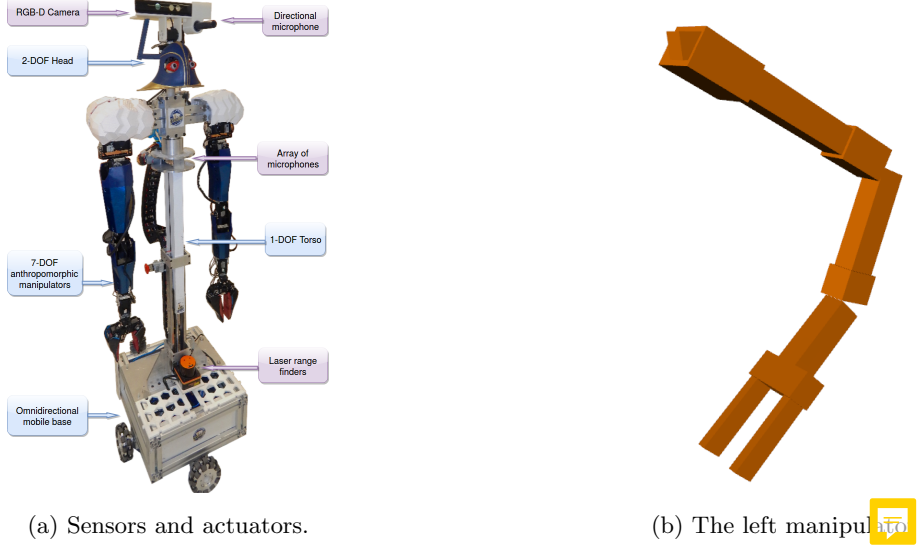
(a) Sensors and actuators.

(b) The left manipulator.

Figure 1: The domestic service robot Justina.

3 Mass estimation with Sliding Mode Observers

3.1 Dynamic Model

From the Langrangian of the manipulator, a dynamic model of the following form can be obtained:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + G(q) + \Delta(q,\dot{q},u) = u \tag{1}$$

where $q \in \mathbb{R}^7$ are the joint angles, $M(q) \in \mathbb{R}^{7\times7}$ is the inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{7\times7}$ is the Matrix of Coriollis forces, $B\dot{q} \in \mathbb{R}^7$ is the vector of friction forces, $G(q) \in \mathbb{R}^7$ is the vector of gravitational forces, u is the input torque, considered as control signal, and $\Delta(q,\dot{q},u)$ is a vector containing all errors due to uncertanties, disturbances and fault signals.

To design a SMO it is necessary to write the model in variable states form. Let $x_1 = [q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ q_7]^T$ and $x_2 = [\dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4\ \dot{q}_5\ \dot{q}_6\ \dot{q}_7]^T$ be the state variables. Then (1) can be written as:

$$\dot{x}_1 = x_2 \tag{2}$$
$$\dot{x}_2 = -M^{-1}(q)\left(C(q,\dot{q})\dot{q} + B\dot{q} + G(q) + \Delta(q,\dot{q},u) - u\right) \tag{3}$$

Equation (3) can also be written in the form:

$$\dot{x}_2 = f(x_1,x_2,u) + \phi(x_1,x_2,u)$$

where $f(x_1,x_2,u) = -M^{-1}(q)\left(C(q,\dot{q})\dot{q} + B\dot{q} + G(q) - u\right) \in \mathbb{R}^7$ is the nominal part and $\phi(x_1,x_2,u) \in \mathbb{R}^7$ contains all terms related to uncertainties, disturbances and fault signals. If the system is correctly identified, and assuming we have no other disturbances than the object being carried, then $\phi(x_1,x_2,u)$ corresponds only to the fault signals, which, in this work, will be caused by the weight of the object being manipulated.

Thus, if we reconstruct the signal ϕ we will be able to estimate the mass of the manipulated object.

3.2 Observer for Disturbance Reconstruction

If a SMO is used to estimate the joint speeds, the unknown term $\phi(x_1, x_2, u)$ in (2)-(3) can be reconstructed by an appropriate filtering of the output error injection term. We used the observer proposed by [9]:

$$\dot{\hat{x}}_1 = \hat{x}_2 + z_1 \tag{4}$$
$$\dot{\hat{x}}_2 = f(x_1, \hat{x}_2, u) + z_2 \tag{5}$$

where z_1 and z_2 are the output error injection terms calculated as

$$z_1 = \begin{bmatrix} z_{11} \\ \vdots \\ z_{17} \end{bmatrix} = \begin{bmatrix} \lambda |q_1 - \hat{q}_1|^{1/2} sign(q_1 - \hat{q}_1) \\ \vdots \\ \lambda |q_7 - \hat{q}_7|^{1/2} sign(q_7 - \hat{q}_7) \end{bmatrix}$$

$$z_2 = \begin{bmatrix} z_{21} \\ \vdots \\ z_{27} \end{bmatrix} = \begin{bmatrix} \alpha sign(q_1 - \hat{q}_1) \\ \vdots \\ \alpha sign(q_7 - \hat{q}_7) \end{bmatrix}$$

In this observer, the sliding surface is given by $\sigma = x_2 - \hat{x}_2$. When the sliding mode is reached, it holds that:

$$\sigma = \dot{\sigma} = \dot{x}_2 - \dot{\hat{x}}_2 = f(x_1, x_2, u) + \phi(x_1, x_2, u) - f(x_1, \hat{x}_2, u) - z_{2_{eq}} = 0$$

Since, $x_2 = \hat{x}_2$, then

$$z_{2_{eq}} = \begin{bmatrix} z_{21_{eq}} \\ \vdots \\ z_{27_{eq}} \end{bmatrix} = \phi(x_1, x_2, u) = \begin{bmatrix} \phi_1(q_1, \ldots, q_7, \dot{q}_1, \ldots, \dot{q}_7, u_1, \ldots, u_7) \\ \vdots \\ \phi_7(q_1, \ldots, q_7, \dot{q}_1, \ldots, \dot{q}_7, u_1, \ldots, u_7) \end{bmatrix} \tag{6}$$

where $z_{2_{eq}}$ is the equivalent output error injection which can be obtained by an appropriate low-pass filtering of z_2.

Noisy and low sampling frequency rates can cause ~~big~~ chattering effects on the estimated states. ~~This is not a~~ problem, ~~since~~ the output error injection will be low-pass filtered, nevertheless, if estimated states are used for a closed-loop control, then actuator signals will have high frequency components that will result in hardware damage. Thus, as it will be later discussed, the SMO is used to estimate the fault signal but, to implement a closed-loop control, an Extended Kalman Filter is used instead.

3.3 Mass estimation

The term $\phi(x_1, x_2, u)$ in equation (3) is a function of input torque, joint positions and joint speeds. The mass of the object could be calculated in any point of the state space, nevertheless, calculations are much simpler if such estimation is made only when the manipulator is in a constant position. If $\dot{q} = \ddot{q} = 0$, then $\phi(x_1, x_2, u) = \phi(q)$ depends only on the gravitational torques caused by the weight of the object being carried. From (3) we can see that disturbance ϕ is a signal of acceleration, not a torque signal. Let $\tau_o = [\tau_o 1, \ldots, \tau_o 7]^T$ be the torque exerted by the manipulated object with mass m_o and weight $w_o = m_o g$
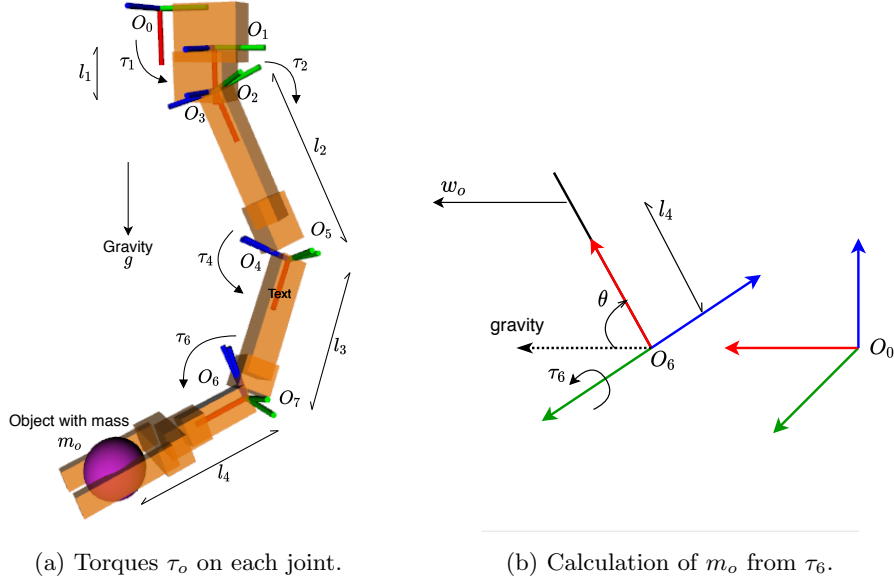
(a) Torques τ_o on each joint.

(b) Calculation of m_o from τ_6.

Figure 2: Variables and frames to calculate m_o.

on each joint. Vector τ_o can be obtained by multiplying signal ϕ by the inertia matrix $M(q)$. Thus, from equation (6):

$$\tau_o = M(q)\phi(q) = M(q)z_{2_eq} \qquad (7)$$

Figure 2a shows torques τ_o on the different joints. Torques are zero where the weight force is applied along the axis of rotation. In figure 2a this is the case for joints O_3, O_5 and O_7.

Torques τ_o are caused by the weight of the object being carried and thus they can be obtained in a form similar to term $G(q)$ in model (1). The mass m_o can be estimated from any component of τ_o, nevertheless, due to the kinematic configuration, it is much easier if we use torque on joint O_6.

Consider the figure 2b. The frame O_6, attached to joint 6, is the frame O_0 translated and rotated. To represent the rotation we used the Roll-Pitch-Yaw angles (ρ, θ, ψ). As it can be seen, the weight $w_o = m_o g$ causes the torque $\tau_6 = -m_o g l_4 \sin\theta$ and the mass of the manipulated object can thus be calculated as:

$$m_o = -\frac{\tau_6}{g l_4 \sin\theta} \qquad (8)$$

where θ is the *pitch* angle of frame O_6 w.r.t. O_0, g is the gravity acceleration, τ_6 is the disturbance torque calculated according to (7) and l_4 is the distance from joint O_6 to the center of mass of the manipulated object. As it will be latter explained, errors in the estimation of l_4 will cause proportional errors in the estimation of m_o.

Note that when $\theta = 0$ it is not possible to estimate m_o because $\theta = 0$ means that the manipulated object is "hanging" from the joint and its weight is not exerting any torque on the joint. There are two possible ways overcome this situation: use another disturbance τ_i or moving the manipulator to a useful

5

position. Second option is more feasible since the relation between w_o and τ_i becomes more complex in the rest of the joints.

3.4 Extended Kalman Filter

Sliding Mode Observer have the great advantage of being robust against disturbances, nevertheless, due to the discontinuos ouput error injection, they can have high frequency components in the estimated states. SMOs are useful for fault reconstruction but, due to the noise and low frequency sampling rate, it is better to use an Extended Kalman Filter (EKF) as part of the closed-loop control strategy.

From (2)-(3), we have the state transition model for the noisy system:

$$\dot{x}_1 = x_2 + \nu_1 \tag{9}$$
$$\dot{x}_2 = f(x_1, x_2, u) + \phi(x_1, x_2, u) + \nu_2 \tag{10}$$

where $\nu \in \mathbb{R}^14$ is a vector of noise signals without temporal correlation, zero mean and covariance matrix $Q \in \mathbb{R}^{14 \times 14}$. Remember that system state variables are the seven joint positions and seven joint speeds.

In this work, since we are measuring the joint positions, our observation model is:

$$z = h(x, u) + \omega = [q_1, \ldots, q_7]^T + \omega \tag{11}$$

where $\omega \in \mathbb{R}^7$ is Gaussian noise without temporal correlation, zero mean and covariance matrix $R \in \mathbb{R}^{7 \times 7}$.

Chosing between the cotinuous or discrete version of the EKF depends on the frequency sampling. In this work we achieved a sampling of 250 Hz, which is much faster than the system dynamics, but too slow for a SMO. Also, since Simulink provide tools for continuos controls, we chose the continuos version of the EKF:

$$\dot{\hat{x}}_1 = \hat{x}_2 + K_1 y_1 \tag{12}$$
$$\dot{\hat{x}}_2 = f(\hat{x}_1, \hat{x}_2, u) + K_2 y_2 \tag{13}$$
$$\dot{P} = FP + PF^T - KRK^T + Q \tag{14}$$

where $y = z - \hat{x}_1$ is the error between the estimated and measured outputs, F is the Jacobian of the state transition model and K is the Kalman Gain calculated as:

$$K = PH^T R^{-1}$$

with $H \in \mathbb{R}^{7 \times 14}$, the Jacobian of the observation model, which, in this case, is the constant matrix:

$$H = \begin{bmatrix} 0 & I_7 \end{bmatrix}$$

Note that in (13) we are using only the nominal part of the system and thus, if $\phi \neq 0$ (i.e., if an object is being manipulated), the estimated states will not converge to the real states. Since estimated states are used for position control, such estimation error will cause a steady state error in the controller. This is in principle undesirable, nevertheless, we can tolerate this error since the main goal of this work is the estimation of the mass, not the performance of the controller. Calculations to estimate m_o are made under the assumption that the manipulator is in a constant position, thus, driving the manipulator to a constant q_f, although slightly different from the desired q_g, is a good enough performance for the control-observation loop.

3.5 Position control

As stated before, equations for estimating m_o are derived under the assumption that the manipulator is in a constant configuration. We implemented a PD+Gravity controller using the EKF estimated states:

$$u = G(\hat{q}) + K_p(q_d - \hat{q}) + K_d(\dot{q}_d - \hat{\dot{q}}) \tag{15}$$

where $\hat{q}$ and $\hat{\dot{q}}$ are the filtered measured positions and EKF-estimated speeds, respectively; $G(\hat{q})$ is the vector of gravity torques of the nominal part of the model (no noise and no fault signals) and

As explained in the previous section, when the manipulator takes an object, the estimated states do not converge to the real ones and then the controller shows an steady state error. Nevertheless, this is tolerable since the goal of the proposed system is not the control performance but the observer performance for the mass estimation.

To improve control performance, a trajectory is planned using a 5th degree polynomial. This trajectory is planned in the joint-space to avoid the necessity of inverse kinematics. Thus, although movement in joint-space is a straight line, movement in the cartesian space will not be a line. Same as with control performance, movement planning is not a goal of this work and it is used only to ease the tuning of control constants.
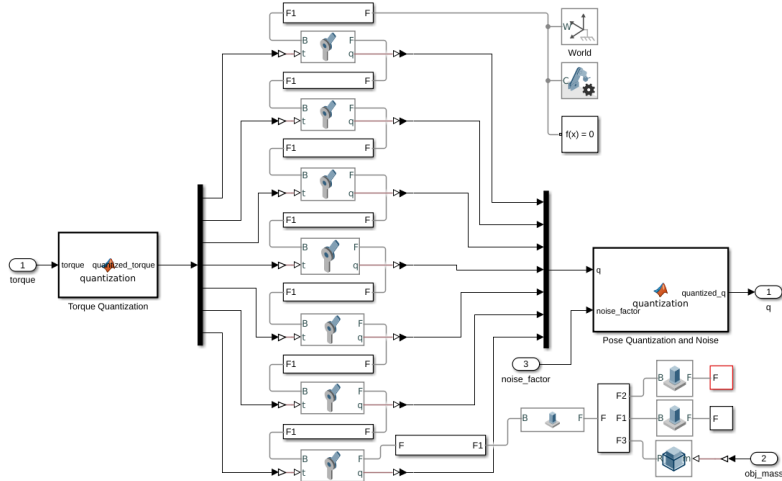
4 Simulink implementation

Figure 3: The resulting model after importing URDF

Implementation of control and observer was made in Simulink mainly using the Robotics, Simscape and ROS Toolboxes. We followed the suggested steps in [5, 6]. To reproduce this proposal in other robots, we recommend to read first such tutorials. In the following subsection we roughly describe this implementation.

Simulator

Most of the tests were performed first in simulation. To such purpose, we used the `smimport` function of ~~the~~ Simscape Toolbox to generate the corresponding rigid body tree. Figure 3 shows the resulting block diagram. As it can be seen, simulation of robot dynamics is separated from control and observation. Communication between manipulator and controller and observer is made via ROS topics in order to keep them transparent to the hardware. This way, switching between real and simulated manipulator is easy.

Quantization and noise

To achieve a simulation more similar to the conditions of the real manipulator, we added a quantization for input torque and position measurements. Dynamixel motors can read position with 12 bit resolution for 360 degrees and can set torque with 10 bit resolution. Torque resolution is not constant since it depends on motor model and battery level. The implemented simulation add this quantization effect. An example of such quantization is shown in figure 4. Also, noise were added to joint measurements. We added a noise equivalent to ∓ 2 bits.

(a) Continuos joint positions   (b) Quantized joint position

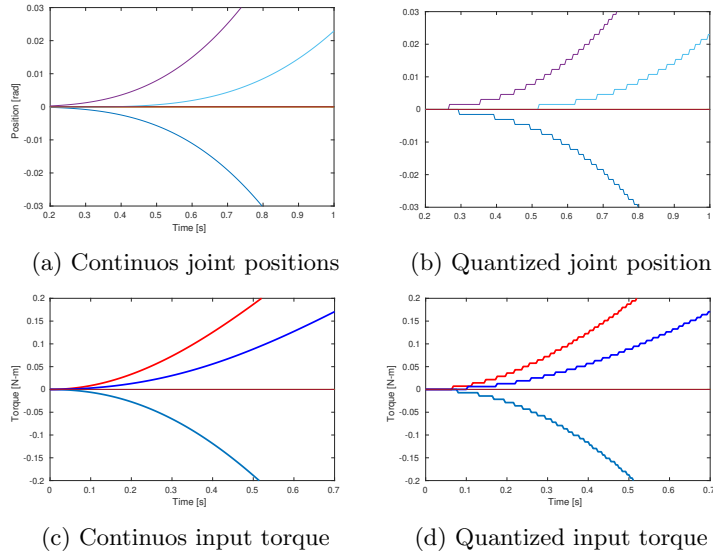(c) Continuos input torque   (d) Quantized input torque

Figure 4: Position and torque quantization

Observation and control

The Sliding Mode Observer is a copy of the system plus an output error injection term, with the form:

$$\dot{\hat{x}}_1 = \hat{x}_2 + z_1 \tag{16}$$

$$\dot{\hat{x}}_2 = -M^{-1}(q)\left(C(q,\dot{\hat{q}})\dot{\hat{q}} + B\dot{\hat{q}} + G(q) - u\right) + z_2 \tag{17}$$

8

The nominal part can be derived from the lagrangian, nevertheless, the analytic form is too complex due to the number of DOF. Instead, we used the numeric solution provided by the Simulink Robotics Toolbox, as shown in figure 5a.

$$\dot{\hat{x}}_1 = \hat{x}_2 + z_1$$
$$\dot{\hat{x}}_2 = -M^{-1}(q)\left(C(q,\dot{q})\dot{q} + B\dot{q} + G(q) - u\right) + z_2$$

$$\tau_o = M(q)\phi(q)$$
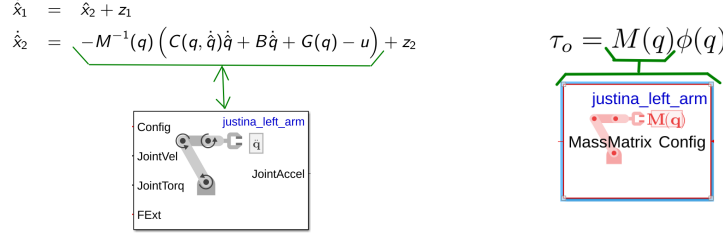
(a) Forward Dynamics block in SMO implementation (b) Mass Matrix block in mass estimation

Figure 5: Implementation using Simulink Blocks

Also, from equation (7), it is necessary to compute the inertia matrix $M(q)$ to obtain the perturbing torques ϕ. Similar to the observer, instead of obtaining the algebraic expression of $M(q)$, we used the numeric calculation provided by the Robotics Toolbox, as shown in figure 5b.

Similar to the SMO, the EKF is a copy of the system plus an output error injection term, but in this case, such term is not discontinuous, but it is a linear gain of the error, with such linear gain calculated to minimize noise effect. As shown in figure 6a, the forward dynamics term in equations (12)-(13) is calculated using the corresponding Robotics Toolbox block.

A ROS node for each task

To ease implementation with the real manipulator and integration with the rest of the subsystems, we separated every task in different ROS nodes: arm dynamics simulation, control, SMO and EKF. Signals are shared via topics and ROS parameters are used for all tuning constants. Figure 7 shows the connections between the different nodes.

$$\dot{\hat{x}}_1 = \hat{x}_2 + K_1 y_1$$
$$\dot{\hat{x}}_2 = f(\hat{x}_1, \hat{x}_2, u) + K_2 y_2$$

$$u = G(\hat{q}) + K_p(q_d - \hat{q}) + K_d(\dot{q}_d - \dot{\hat{q}})$$
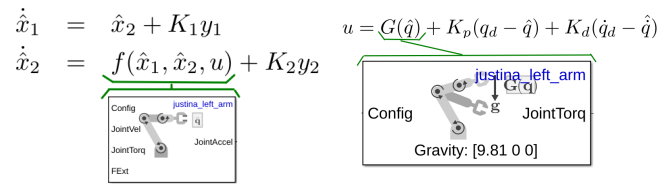
(a) Simulink implementation of the EKF (b) Simulink implementation of PD+ control

Figure 6: Control and observation using Simulink blocks
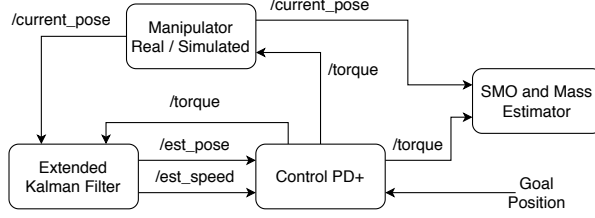
Figure 7: Block diagram of ROS-nodes communications

5 Simulation results

5.1 Description of the experiment

Mass estimation was tested moving the manipulator to several positions commonly used for grasping objects. Figure 8 shows these predefined positions. It is worth to note that mass estimation will work in any configuration as long as wrist pitch θ (see equation (8)) is different from zero.

Frequency sampling was set to 250 Hz. This value is the frequency achieved with the real manipulator. Dynamixel servomotors were configured to 1 Mbps of baudrate and we have nine motors in the manipulator: seven motors for the seven degrees of freedom, and two more for the manipulator. Considering the total number of bytes to be sent and received, the required time is in the order of 1 ms, nevertheless, smallest configurable USB latency in Ubuntu is 1 ms, which needs to be added to the time for sending and receiving RS485 data. 4 ms was chosen as approximately twice the time required to send torques and read positions from all motors. This frequency was set both for tests with real and simulated manipulator.
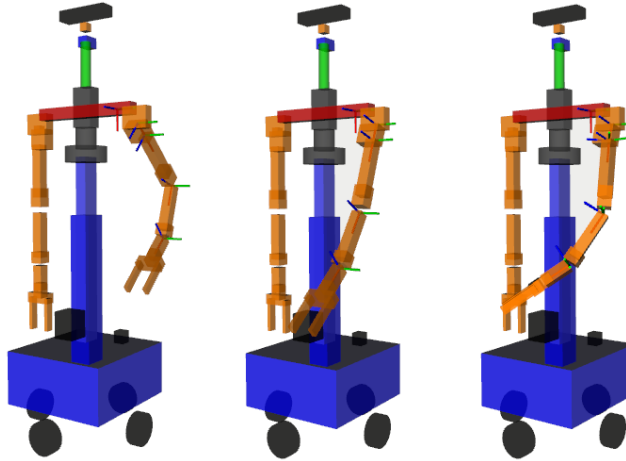

Figure 8: Positions used for testing masss estimation

Experiment for mass estimation was made simulating an object with mass $m = 0.25$ located at the center of the gripper. We used the constants shown in table 1, where I_{14} and I_7 represent identity matrices of orders 14 and 7

SMO	EKF	Control	Mass Est
$\lambda = 6.0$	$Q = 0.003I_{14}$	$K_p = 2.5$	$l_4 = 0.21$
$\alpha = 4.2$	$R = 0.001I_7$	$K_d = 0.5$	

Table 1: Constants for control and observation

respectively. To filter the output error injection term of the SMO we used a 4th degree Butterworth Low-Pass filter with cutoff frequency of 1 Hz.

5.2 Mass estimation

Figure 9 shows the results for the estimation and filtering of the angular position while moving the arm to one of the predefined positions. As it can be observed, the measured position is noisy and does not converges to the desired position, nevertheless, this is an expected behavior. As explained before, the objective of this work is the mass estimation of the manipulated object, not the performance of the controller. In the presence of the fault signal (an object in the end effector), the controller shows an steady state error. Comparing the estimated positions between the SMO and the EKF, we can see that the SMO-estimated positions perfectly track the measured positions, since in general, SMOs are designed to be robust against faults and disturbances, nevertheless, SMO-estimated positions show the problem of chattering. Instead, EKF-estimated positions show an steady-state error but do not show chattering nor noise, making them more suitable for implementing the PD+ control.

Figure 10 shows the results for the estimation of the angular speeds. Although in the real manipulator we don't have a measurement of the joint speed, we included the simulated speed for testing purposes. As it can be seen, current speed does not converge to the desired one. Same as the angular position, this error is due to the simplicity of the controller. Similar to the positions, the SMO-estimated speeds are nearer to the current speeds but with the problem of big chattering. Instead, the EKF-estimated speeds show a big error but without noise, which makes them better to be used in the controller.

Finally, figure 11 shows the resulting estimated mass. As explained in section 3.3, in this work we make the mass estimation only when the manipulator is in a constant position. Ideally, estimations should be made only when $\dot{q} = 0$, nevertheless, this is a hard condition. Instead, we set estimated mass to zero if $\|\dot{q}\| > 0.5$, that's why first seconds in figure 11 are zero. As soon as the arm stops, the estimated value converges to 0.25 kg, the actual mass of the object.

5.3 Sources of estimation errors

There are two possible causes of errors in the estimation: uncertanties in the physical parameters (link masses and inertia, joint frictions, etc) and the position of the object along the gripper. In this work it is assumed that the system is correctly identified, nevertheless, sometimes this assumption could not be hold, specially in complex systems such as the 7-DOF manipulator. In order to analyze the effect of parameter uncertainties in the estimation, we ran simulations with wrong parameter values. That is, the term $f(x_1, \hat{x}_2, u)$ in (16)-(17) is calculated with the wrong parameters but measurements q come from the
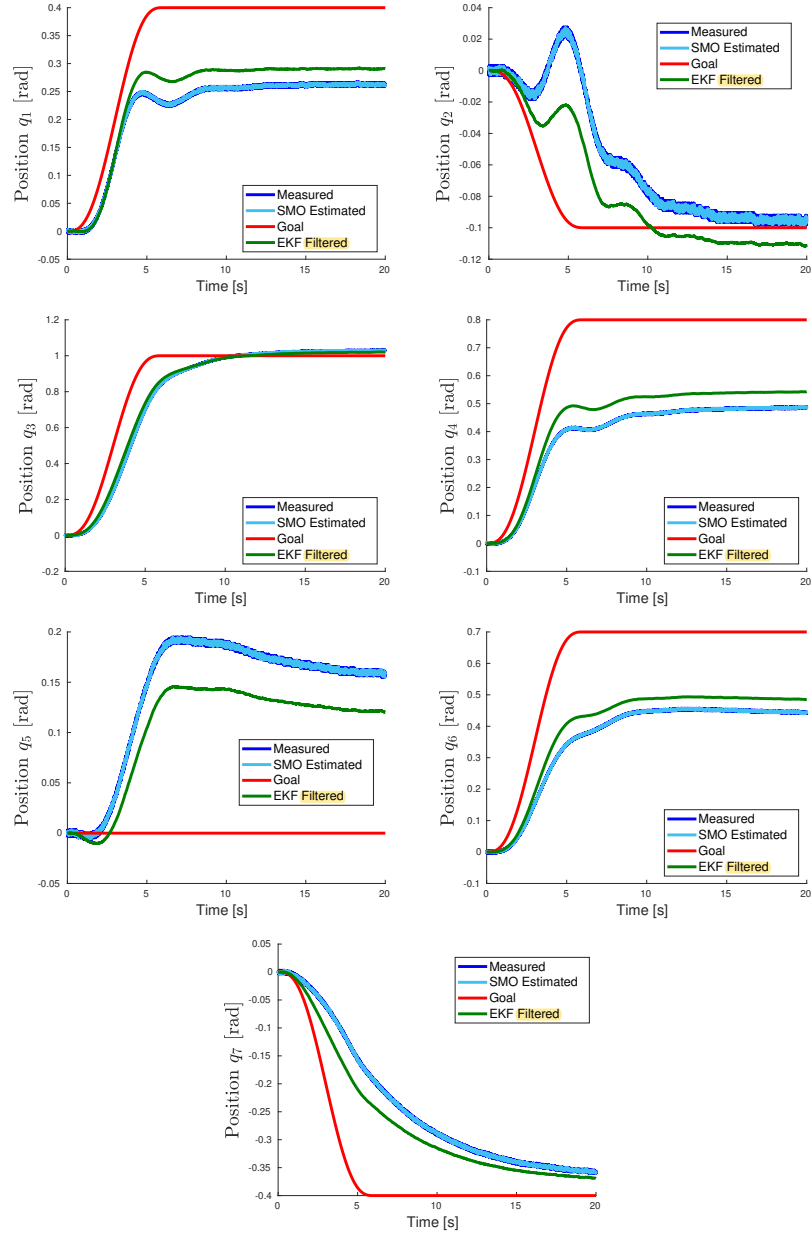
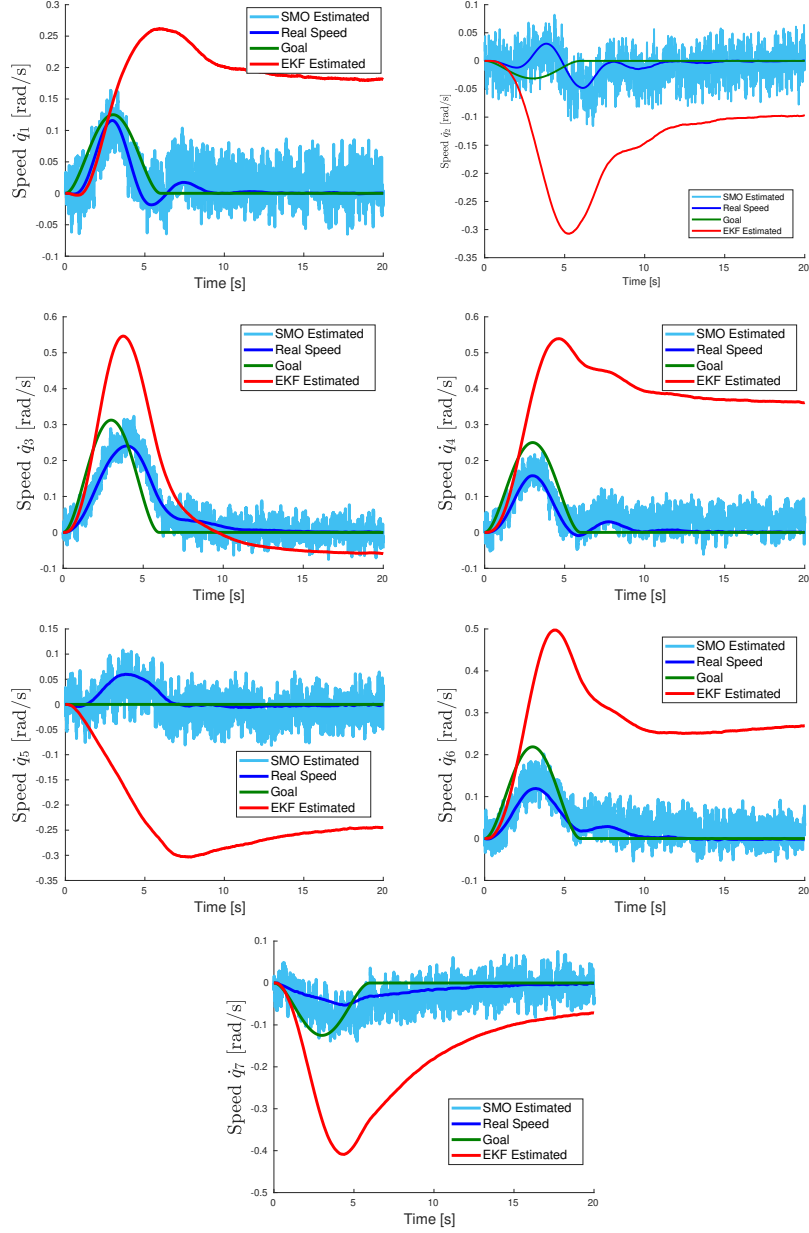Figure 9: Comparison of measured, goal and estimated positions.

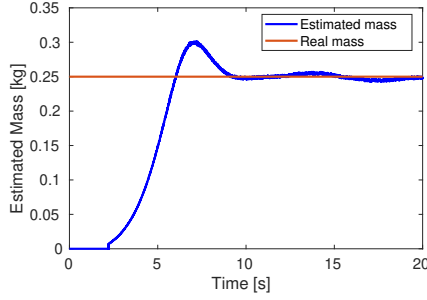Figure 10: Comparison of simulated, goal and estimated joint speeds.
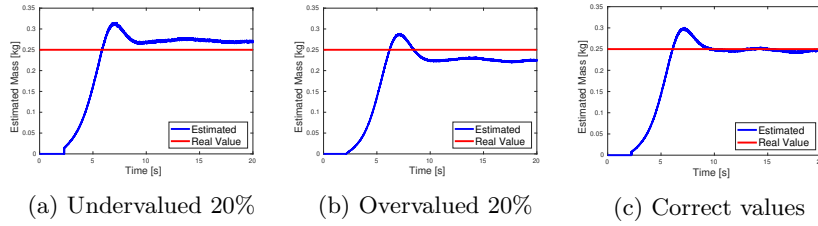
Figure 11: Mass estimation results.

(a) Undervalued 20%   (b) Overvalued 20%   (c) Correct values

Figure 12: Effect of parameter uncertanties

system simulated with the correct values. Figure 12a shows the estimation results using link masses and inertia with values 20% smaller than the nominal ones. As it can be seen, the system overestimates the mass value. The torque exerted on each joint is caused by the link mass and the manipulated object. If such link mass is smaller, then the exerted torque will be attributed to the manipulated object. The contrary effect happens when the physical parameters are overestimated, as shown in figure 12b.
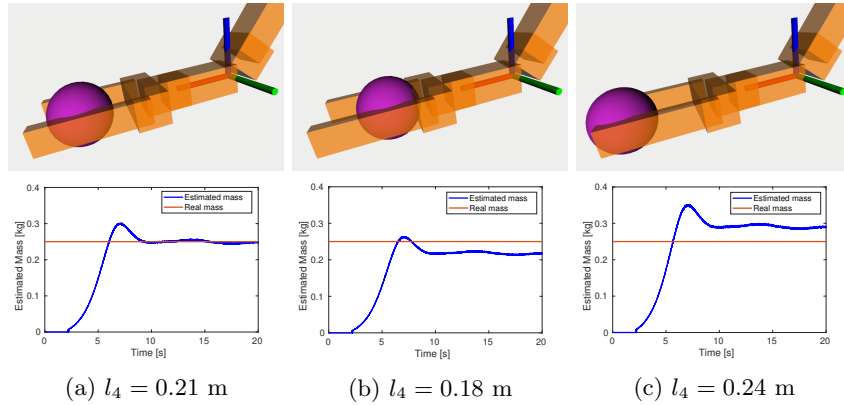

(a) $l_4 = 0.21$ m   (b) $l_4 = 0.18$ m   (c) $l_4 = 0.24$ m

Figure 13: Effect of object position along gripper in mass estimation

According to equation (8), we need to know the distance l_4 for a correct mass estimation. This distance depends on the object position along the gripper. We always assume that the object is gripped in the center of the final effector,

14

nevertheless, there could be a small error in this assumption. The error in the estimated mass will be proportional to the error in the estimation of l_4. Figure 13 shows the estimated values for three different values of l_4. As it can be seen, the greater the distance l_4, the greater is the torque exerted on the joint with the same object mass and a greater value of mass is estimated. At this stage of the project, we don't have a way to determine the position of the object along the gripper, nevertheless, for the goal application of this project, the errors in estimation are tolerable.

6 Reusability

To reproduce the results presented in this work, the following rough steps should be followed:

- Get the URDF file of the manipulator and import it using the Simulink Robotics Toolbox. If the full robot URDF is available, the manipulator part should be isolated in order to make calculation faster.

- Identify the joint with the following features:
    - The joint must have an axis of rotation perpendicular (or nearly perpendicular) to the gravity vector.
    - The joint should be the nearest one to the end effector with the previous feature.

    And get the distance from the previous joint to the center of the end effector.

- Implement SMO and use equation (8) to estimate object mass.

To illustrate these steps, we will reproduce the results using the description of Katana Manipulator (`https://wiki.ros.org/katana_driver`).

Get and import the URDF

We imported the URDF from the GitHub Katana repository. Most of robot descriptions are given using `xacro`, but the ROS Xacro package can be used to get the corresponding URDF, since Robotics and Simscape Toolbox can only import URDF files. We used Simscape to import the URDF and we added the corresponding ROS publisher and subscriber for measured joint positions and torques respectively. Figure 14 shows the manipulator as imported by Simscape, with ROS publishers and subscribers and the system as displayed in the Mechanics Explorer.

Chosing the correct joint to estimate mass

In this work we used the wrist pitch of our manipulator to estimate object mass. Using the observer (16)-(17), we reconstruct the fault signals $\phi(x_1, x_2, u)$, after that, using equation (7) we get the torques exerted on each joint and finally, using torque on joint 6 (wrist pitch) and equation (8), we get the estimated mass of the manipulated object. The joint of the wrist pitch fulfills the two
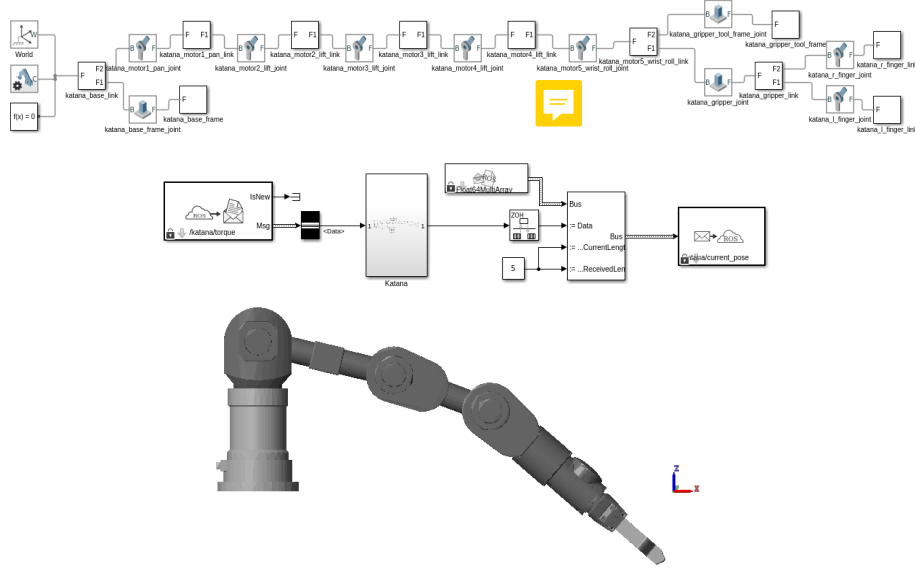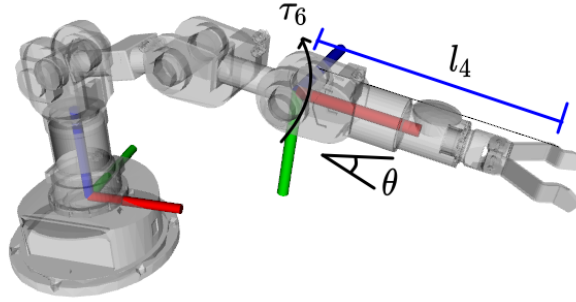
Figure 14: Importing Katana URDF to Simulink

Figure 15: Katana equivalent variables for mass estimation

features previously mentioned: it is the nearest joint to the end effector whose rotation axis is perpendicular to the gravity vector.

In the case of the Katana manipulator, the joint with these features is `katana_motor4_lift_link` and thus, the fault torque associated to this joint will be used in equation (8). Figure 15 shows the equivalent variables in the Katana Manipulator for mass estimation. Angle θ can be calculated using the Euler Angles of the rotation from `katana_base_frame` to `katana_motor4_lift_link`. Robotics Toolbox provides the needed blocks to calculate this angle. Distance l_4 can be obtained from URDF or by directly measuring it.

SMO implementation and mass estimation

Once we identified the ideal joint for mass estimation, we imported the URDF file to get a Rigid Body Tree using the `importrobot` function of the Robotics Toolbox. We implemented the SMO and mass estimator using the Forward
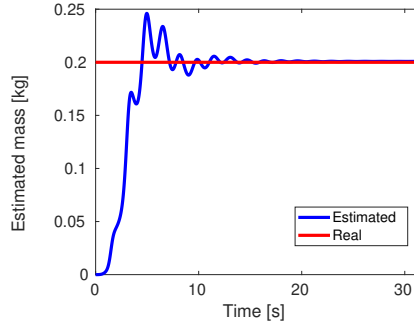
Figure 16: Mass estimation results with the Katana Manipulator

Dynamics and Mass Matrix block as shown in figure 5b (see explanation in section 4). We simulated an object with a mass of 200 g. To simplify simulations, we did not implemented a control but we used only a constant torque applied to the joints. Figure 16 shows the resulting estimation.

7 Conclusions

In this work we presented a method for estimating the mass of the object being manipulated in a domestic service robot. The estimation is based on model based fault reconstruction techniques using sliding mode observer. The implementation of model-based algorithms could be very complicated in complex systems such as the 7 DOF manipulator used in this proposal; nevertheless, the use of Simulink Robotics, ROS and Simscape Toolboxes, which provide numerical solutions to the system dynamics, allowed the implementation of such model-based techniques. We presented simulation results where we show the effectiveness of our proposal. Also, we discussed the cases where this approach can lead to a wrong estimation. To show the possible reusability of our proposal, we replicated some of the results in a simulated Katana manipulator.

Further tests should be made for an implementation in the real manipulator. For example, in this work we assumed that only linear friction is present in the motor, nevertheless, Dynamixel motors present also non linear frictions, which must be modeled and identified for a correct performance. Also, better controllers can be implemented to reach faster the steady state. A shorter settle time will result in a faster mass estimation.

As a future work, different techniques can be implemented. Another model-based option is the Extended Kalman Filter. As shown in section 5, the EKF estimated speeds have a steady state error due to the fault signal. This estimated speeds can also be used to estimate the manipulated object mass. On the other hand, model-free techniques can also be implemented, such as neural networks. Input torques and measured positions can be taken as input data to train a NN whose output is the estimated mass.

Finally, this proposal will be integrated with the rest of subsystems of our domestic service robot in the context of the Robocup@Home tests.

References

[1] H. Alwi, C. Edwards, and C. P. Tan. *Fault detection and fault-tolerant control using sliding modes*. Springer Science & Business Media, 2011.

[2] J. Davila, L. Fridman, and A. Levant. Second-order sliding-mode observer for mechanical systems. *IEEE transactions on automatic control*, 50(11):1785–1789, 2005.

[3] S. X. Ding. *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. Springer Science & Business Media, 2013.

[4] L. Fridman, A. Levant, et al. Higher order sliding modes. *Sliding mode control in engineering*, 11:53–102, 2002.

[5] MathWorks-Students-Competitions-Team. Designing robot manipulator algorithms (https://github.com/mathworks-robotics/designing-robot-manipulator-algorithms). GitHub. Retrieved April 6, 2020., 2020.

[6] MathWorks-Students-Competitions-Team. Trajectory planning for robot manipulators (https://github.com/mathworks-robotics/trajectory-planning-robot-manipulators). GitHub. Retrieved April 6, 2020., 2020.

[7] R. Memmesheimer, I. Mykhalchyshyna, N. Wettengel, T. Evers, L. Buchhold, P. Schmidt, N. Schmidt, I. Germann, M. Mints, G. Rettler, C. Korbach, R. Bartsch, I. Kuhlmann, T. Weiland, and D. Paulus. Robocup 2019 - homer@unikoblenz. https://github.com/RoboCupAtHome/AtHomeCommunityWiki/wiki/files/tdp/2019-opl-homeratunikoblenz.pdf, 2019.

[8] J. Savage, A. LLarena, G. Carrera, S. Cuellar, D. Esparza, Y. Minami, and U. Peñuelas. Virbot: a system for the operation of mobile robots. In *RoboCup 2007: Robot Soccer World Cup XI*, pages 512–519. Springer, 2008.

[9] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant. *Sliding mode control and observation*. Springer, 2014.

[10] M. Van Der Burgh, J. Lunenburg, L. van Beek, J. Geijsberts, L. Janssen, S. Aleksandrov, K. Dang, van Rooy H., A. Hofkamp, D. van Dinther, A. Aggarwal, and M. van de Molengraft. Tech united eindhoven @home 2019 team description paper. https://github.com/RoboCupAtHome/AtHomeCommunityWiki/wiki/files/tdp/2019-dspl-techunited_eindhoven.pdf, 2019.

[11] S. Wachsmuth, D. Holz, M. Rudinac, and J. Ruiz-del Solar. Robocup@ home-benchmarking domestic service robots. In *AAAI*, pages 4328–4329, 2015.

[12] M. Williams, B. Johnston, S. Pfeiffer, J. Vitale, J. Clark, L. Kang, D. Ebrahimian, S. Leong, R. Billingsley, M. Tonkin, S. Herse, S. Alam, S. Gudi, and S. Ojha. Uts unleashed! 2019 robocup@home social spl. https://github.com/RoboCupAtHome/AtHomeCommunityWiki/wiki/files/tdp/2019-sspl-uts_unleashed_.pdf, 2019.