Noname manuscript No.
(will be inserted by the editor)

Weight Estimation in Manipulation Tasks of Domestic Service Robots using Fault Reconstruction Techniques

Received: date / Accepted: date

Abstract Object manipulation is a key capability in domestic service robots (DSR). Most of works in this area focus either in locating the target object or planning movements, but there is an underlying assumption that there is information about the physical properties of the object, such as weight, surface friction, etc. Estimation of the weight of the grasped object has not been widely addressed. In this work we propose to apply fault reconstruction techniques to estimate the mass of the grasped object. Considering the manipulator without load as the nominal system, and the weight of the object as a fault signal, we can reconstruct such signal using a Sliding Mode Observer. Since our proposal needs to drive the manipulator to certain configurations, we also implemented an Extended Kalman Filter and a PD+ control for position control. We tested our proposal in simulation using the model of the manipulator of our domestic service robot and we also discuss the possible sources of error. To show the reusability of our proposal, we also tested our system with a simulated model of the Neuronics Katana. Finally, we state our conclusions and sketch the future work.

Keywords Object Manipulation · Service Robots · Sliding Mode Observers · Fault Reconstruction

1 Introduction

Domestic Service Robots (DSR) are robots intended to assist humans in everyday tasks in environments such as homes or offices [13]. To achieve their goal, domestic service robots require several capabilities, such as planning, human-robot interaction, navigation, adaptive behavior and object recognition and manipulation [44]. Evaluating DSR performance is challenging due to the wide range of involved technologies and the non-standard environment

Address(es) of author(s) should be given

the robot interacts with [45]. Robocup@Home is a competition for benchmarking DSR where performance is graded through a series of tasks designed to evaluate both single capabilities and system integration.

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 the inverse kinematics of 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 DSRs, they are unable to distinguish between empty and full cans, since object recognition commonly rely only on visual information, sometimes enriched with depth information. On the other hand, manipulation algorithms commonly assume there are control laws to move the manipulator along the planned trajectories. Such control laws are commonly tunned to perform using the specific manipulator of the robot. Nevertheless, when the robot grasps an object, the physical parameters change and thus the tuned constants (either PID or other techniques) could not achieve the same performance.

We consider that estimating the weight of the grasped object can be useful to improve manipulation performance in DSR. This improvement can be achieved from low-level controllers (by adjusting control constants according to the estimated weight) to high-level planning (by allowing the robot to distinguish, for example, full and empty containers and planning actions accordingly). There are many works where object recognition and manipulation is improved by incorporating information different from the visual one. For example, the work of [12] incorporates information from navigation to the object recognition process and [18] uses a smart environment (with RFID tags) to improve the general performance of manipulation tasks of a DSR.

Our proposal consists on applying model-based fault reconstruction techniques [11] to estimate the weight of the object being carried by the robot. If we consider the manipulator without load 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 [36]. SMO also provide the ability to reconstruct the fault signal by filtering the so called output error injection term [3]. With the estimation of the manipulated object mass, we expect to improve the manipulation performance in DSR.

This document is organized as follows: In section 2 we present the related work and emphasize our contribution. In section 3 we describe in detail our proposal, starting with SMO, the service robot we tested with, the observer for fault reconstruction and the algorithm for estimating the weight of the grasped object. Section 4 describes the control law we used to perform the

movements. In section 5 we describe the tests we made in simulation and the results we obtained. In this section we also discuss the possible sources of error in the estimation process. In order to show the reusability of our proposal, in section 6 we reproduce the results using a different manipulator: the Neuronics Katana. Finally, in section 7 we state our conclusions and sketch the future work.

2 Related work

Object manipulation in DSR involves recognizing (and locating) the target object, and planning movements to grasp it (which also include the design of control laws to perform such movements). Object recognition in DSR was commonly addressed with feature-based algorithms such as SIFT and SURF, examples of this approaches are [39] and [37]. The work of [23] reviews object recognition techniques based on features with specific applications to mobile robots. Since domestic service robots are commonly equipped with pan-tilt units, active vision approaches are also widely used. More recently, neural network-based techniques have become popular with the release of ready-to-use tools such as TensorFlow [1], OpenVino and Yolo [32]. Examples of robots using this Neural Network techniques are [26], [43] and [6]. In [15] authors make a survey about several works using neural network-based techniques. All these works perform object recognition using only visual information. To the knowledge of authors, estimation of object's physical properties has not been addressed as part of the manipulation task in DSR.

Research on manipulation for DSR has mostly focused on motion planning, either for obtaining trajectories, avoiding obstacles or to calculate the best grasping configuration [4]. There are many works that use data-driven techniques to find the best grasping point [5,33], although more recently, deep learning techniques have achieved larger advancements [28]. Works for collision avoidance include the self-collision avoidance, such as the works of [27] and [31], and collision avoidance with the environment, such as the works of [10] and [29]. Collision avoidance has also been tackled using reinforcement learning [46] and imitation of the human experience [17]. In [15] authors make a review of the current advances in manipulation for service robots. In a lower level of execution, after a movement is planned, the manipulator requires of control laws that ensure the movement will be performed as planned. In this area, there are many works using classic model-based techniques, such as [41, 35, 22] and [16], and also using neural network approaches [20, 21, 19]. In most of the presented works, an implicit assumption is the availability of information about physical properties such as surface frictions or weights.

On the other hand, fault reconstruction techniques have been applied to robot manipulators mostly to detect and isolate faults on motor actuators and sensors [30, 8, 47]. Fault detection and isolation, from the control theory approach, is commonly achieved by the implementation of residual generators [2]. A common way to design such residual generators is through observers

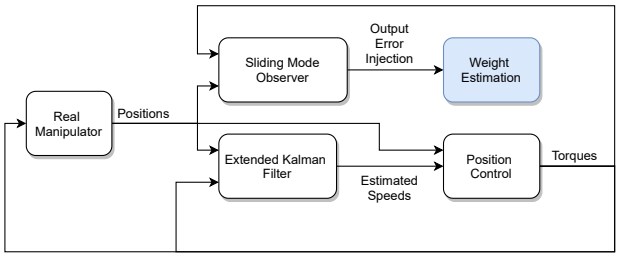


Fig. 1: Block diagram of the proposed full system.

that are sensitive to the fault signals. Sliding mode observers (SMO) 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 type of observers can be used to reconstruct faults or disturbance signals through an appropiate filtering of the so-called injection output error [36,9]. There are many works using SMO for fault diagnosis in robot manipulators, such as [7,40] and [14]. Although fault reconstruction in manipulators using SMO is a common approach, it has not been used to estimate the weight of the grasped object.

As stated in [38], payload is a strong factor in the likelihood of failure of a robot manipulator and it has high impact on the general robot success. That is why in this work we consider important to estimate the weight of the grasped object. In the following section, we describe in detail how we estimate such object's weight by considering the manipulator as a faulty system and using a SMO to reconstruct the fault signal.

3 Weight estimation with Sliding Mode Observers

The system we propose consist on a SMO to estimate the weight of the grasped object and a control block to drive the manipulator to a useful position. It is important to note that the main goal of this work is the observer, nevertheless, since we perform the estimation only in a certain configurations, we also include a PD+ control. The servomotors we use have speed sensors but readings are noise and with low reliability, thus, we also implemented an Extended Kalman Filter (EKF) to estimate the joint speeds. We late explain the reasons to not use the SMO for joint speed estimation. Summarizing, our proposal consist on a SMO for fault reconstruction, an EKF for joint speed estimation and a PD+ module for position control. Figure 1 shows a block diagram with the full system.

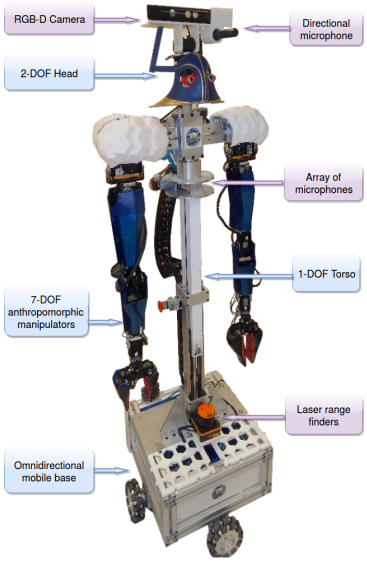


Fig. 2: The domestic service robot Justina.

3.1 Manipulator's Dynamic Model

Justina is a domestic service robot built at the Biorobotics Laboratory of the National Autonomous University of Mexico and developed under the ViRBot architecture [34]. This robot and its predecessors have been participating in the Robocup@Home league [42] since 2006 performing several tasks such as cleaning a table, serving drinks, and several other tasks that humans ask for. Figure 2 shows robot Justina and its sensors and actuators. In this work we used a simulated version of the left arm of our domestic service robot.

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 [36]:

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

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:

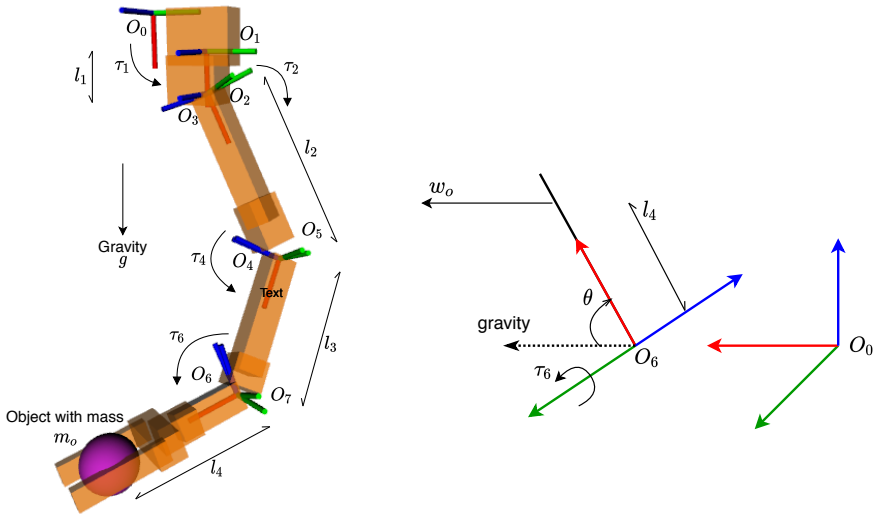
$$\sigma = \dot{\sigma} = \dot{x}_2 - \dot{\hat{x}}_2 = f(x_1, x_2, u) + \phi(x_1, x_2, u) - f(x_1, \hat{x}_2, u) - z_{2_{eq}} = 0$$

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 chattering effects on the estimated states. To mitigate this effect on mass estimation, the output error



(a) Torques τ_o on each joint.

(b) Calculation of m_o from τ_6.

Fig. 3: Variables and frames to calculate m_o.

injection will be low-pass filtered. However, 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 $w_o = m_o g$ on each joint. Vector τ_o can be obtained by multiplying signal ϕ by the inertia matrix $M(q)$. Thus, from equation (6):

$$\tau_o = M(q)\phi(q) = M(q)z_{2_e q} \tag{7}$$

Figure 3a shows torques τ_o on the different joints. Torques are zero where the weight force is applied along the axis of rotation. In figure 3a 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 3b. 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:

$$m_o = -\frac{\tau_6}{g l_4 \sin\theta} \tag{8}$$

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 to overcome this situation: use another disturbance τ_i or moving the manipulator to a useful position. The second option is more feasible since the relation between w_o and τ_i becomes more complex in the rest of the joints.

4 Position control

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 $\nu \in \mathbb{R}^1 4$ is a vector of noise signals without temporal correlation, zero mean and covariance matrix $Q \in \mathbb{R}^{14\times14}$. 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 $R \in \mathbb{R}^{7\times7}$.

Chosing between the continuous 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:

$$H = [0 \quad I_7]$$

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.

4.1 PD+ 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:

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

where $\hat{q}$ and $\dot{\hat{q}}$ are the EKF estimated positions and 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 type of trajectory allows to set initial and final position, speed and acceleration, which is ideal for the manipulation tasks. It is important to note that since the trajectories are calculated in joint space, the cartesian movement of the end effector doesn not result in a straight line. Although algorithms to plan movement task for manipulators are out of the scope of this research, we used this approach to simplify the tuning of the control constants for experimentation.

5 Simulation results

Implementation of control and observer was made in Simulink mainly using the Robotics Systems Toolbox, Simscape and ROS Toolbox. We followed the suggested steps in [24,25]. Simulation of robot dynamics is separated from control and observation. Communication between manipulator, 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 expected to be easy. In the following subsection we roughly describe this implementation.

5.1 Simulink Implementation

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 adds this quantization effect. An example of such quantization is shown in figure 4. Also, noise was added to joint measurements. We added a noise equivalent to ± 2 bits.

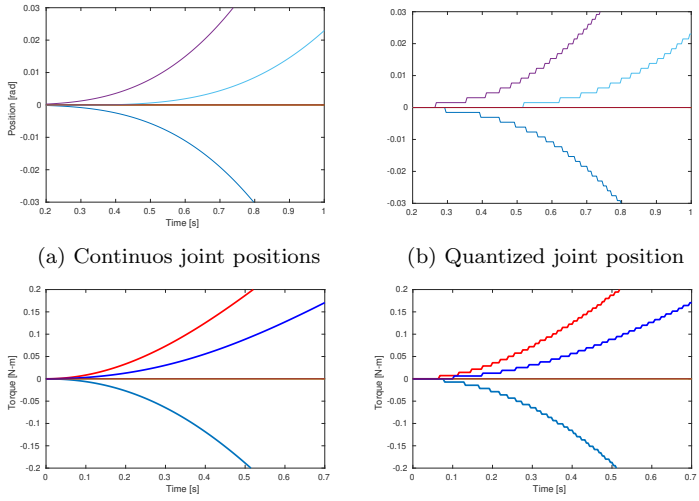
Observation and control

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

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

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

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



(a) Continuos joint positions

(b) Quantized joint position

(c) Continuos input torque

(d) Quantized input torque

Fig. 4: Position and torque quantization

numeric solution provided by the Inverse Dynamics block from the Simulink Robotics System Toolbox library. 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 System Toolbox.

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. The forward dynamics term in equations (12)-(13) is calculated using the corresponding Robotics System Toolbox block. Figure 5 shows the block diagram of the full system as implement in Simulink.

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 6 shows the connections between the different nodes and the ROS topics they communicate through.

5.2 Description of the experiment

Mass estimation was tested moving the manipulator to several positions commonly used for grasping objects. Figure 7 shows these predefined positions. It

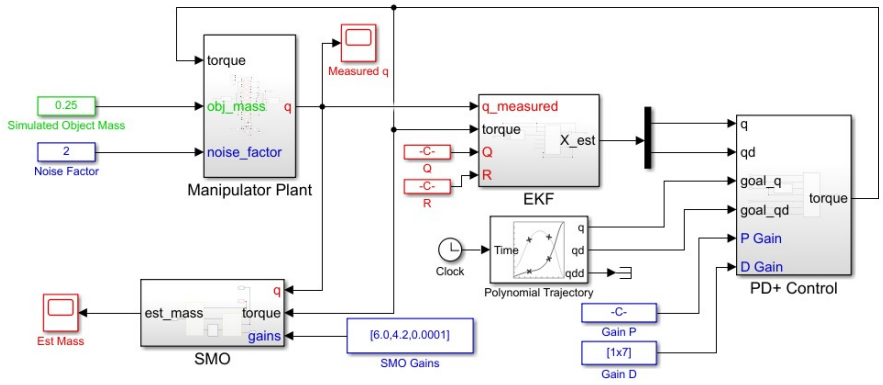


Fig. 5: Block diagram of the full system implemented in Simulink

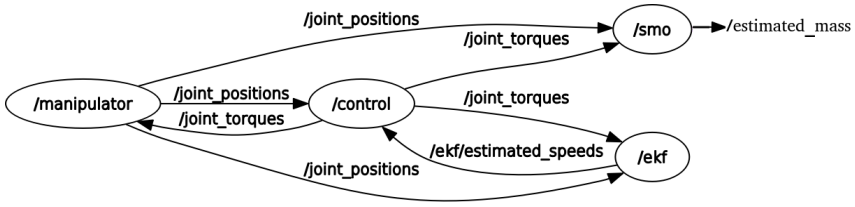


Fig. 6: Graph diagram of ROS-nodes communications

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. In a real application, it is better to estimate the mass only when $|\theta| > \theta_T$ where θ_T is a threshold large enough to avoid large errors due to the division by a very small value of θ. In this work, we used $\theta_T = 0.2$.

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.

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

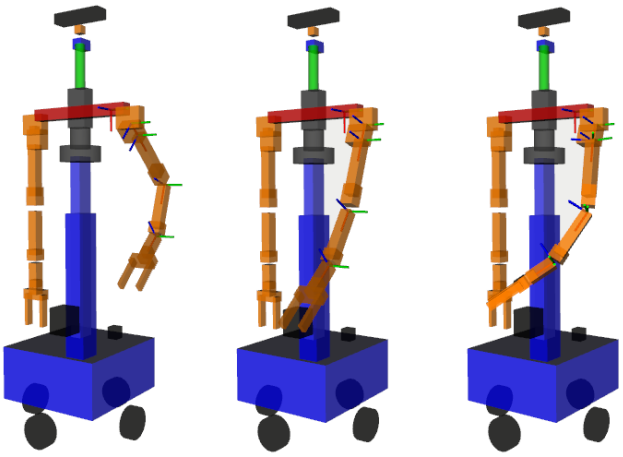


Fig. 7: Positions used for testing masss estimation

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.

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

5.3 Mass estimation

Figure 8 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 converge 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 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 9 shows the results for the estimation of the angular speeds. Although in the real manipulator we don't have a measurement of the joint

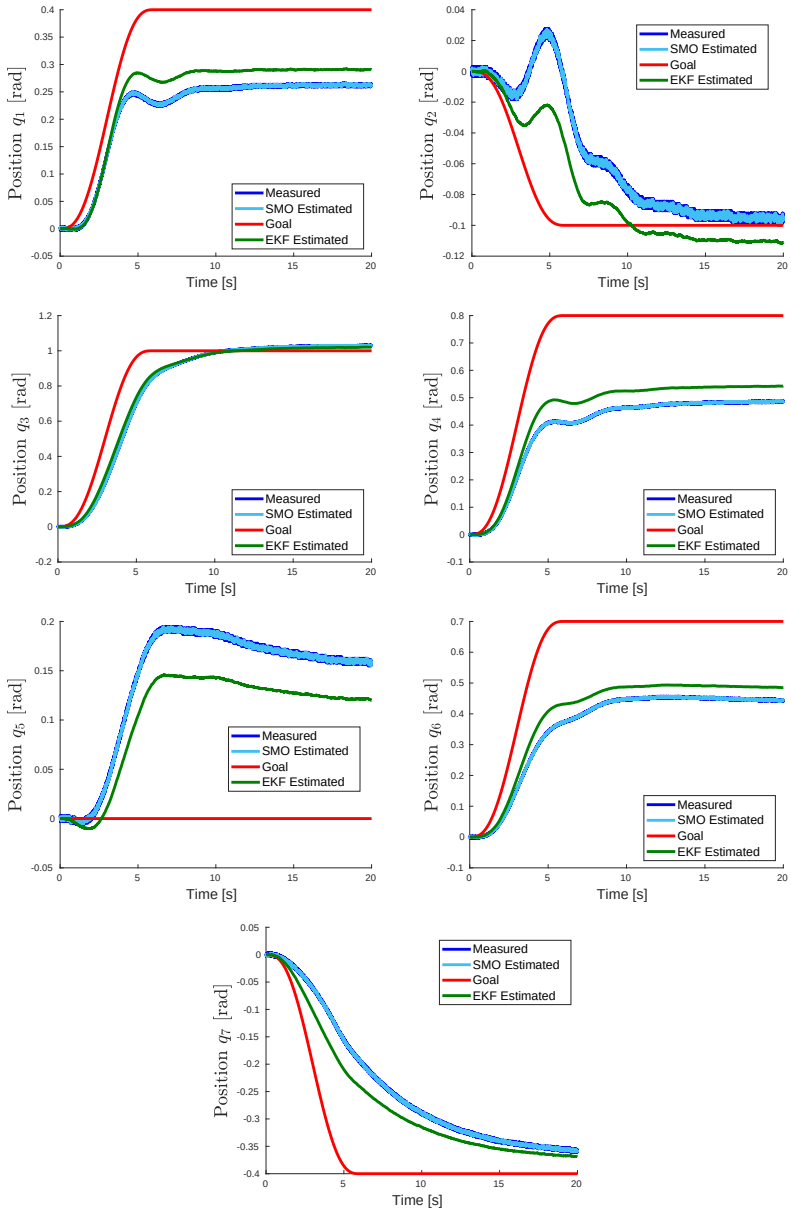


Fig. 8: Comparison of measured, goal and estimated positions.

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 10 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 is why in figure 10 the estimation is equal to zero for the first few seconds. As soon as the arm stops, the estimated value converges to 0.25 kg, the actual mass of the object.

5.4 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 held, 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 system simulated with the correct values. Figure 11a 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 11b.

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 end effector, 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 12 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.

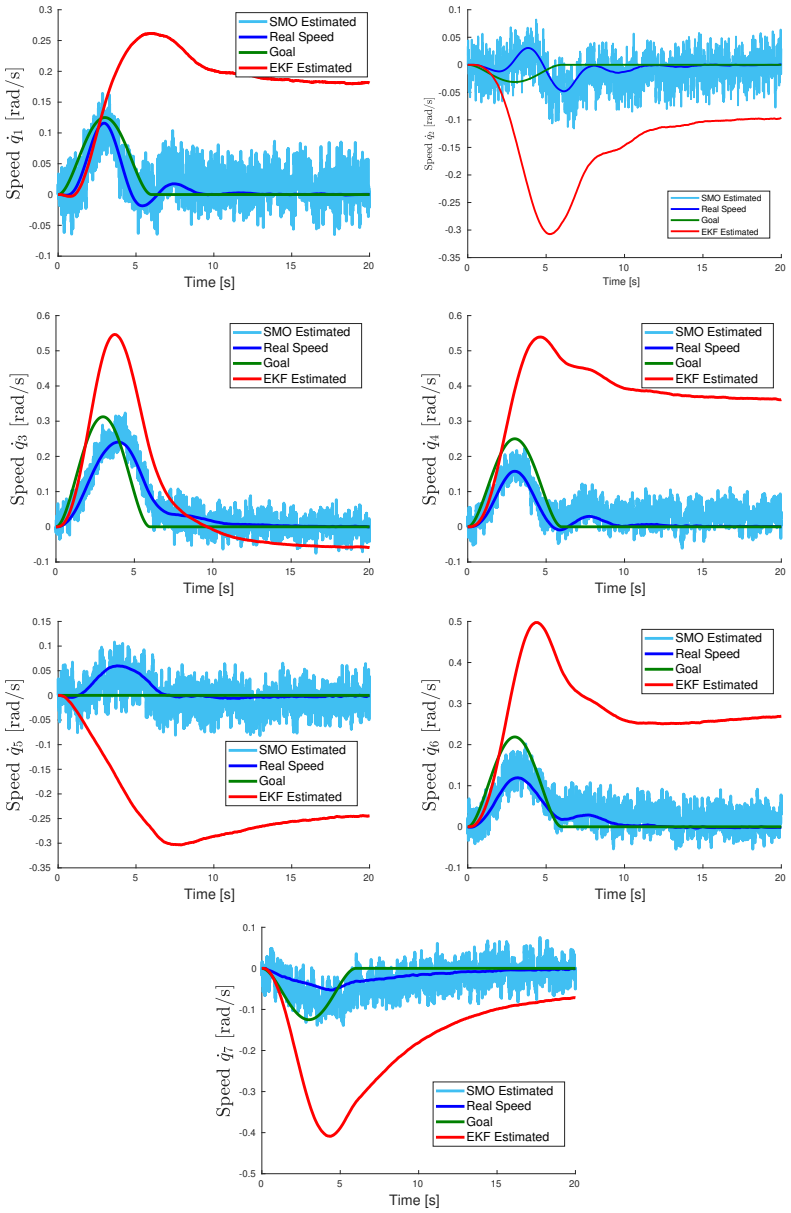


Fig. 9: Comparison of simulated, goal and estimated joint speeds.

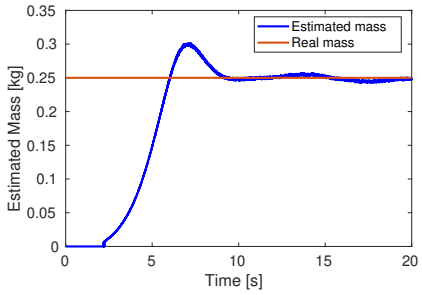
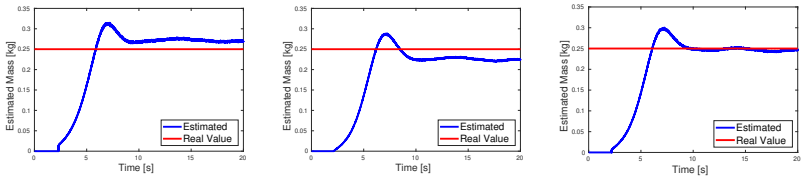
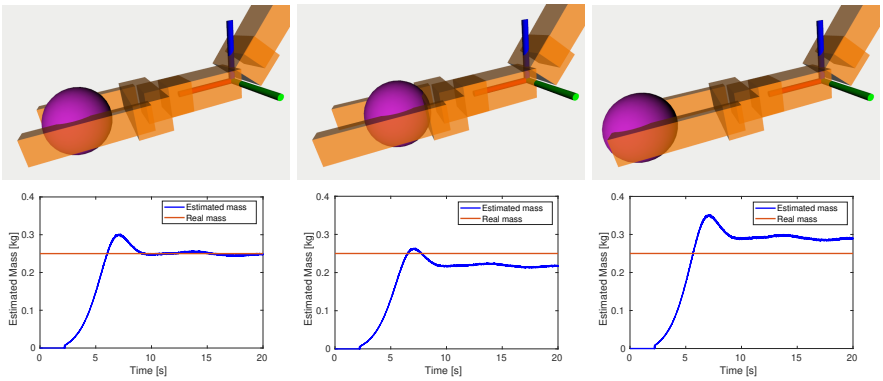


Fig. 10: Mass estimation results.



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

Fig. 11: Effect of parameter uncertanties



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

Fig. 12: Effect of object position along gripper in mass estimation

6 Reusability

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

1. Get the URDF file of the manipulator and import it using the Simulink Robotics Toolbox.
2. Identify the joint with the following features:

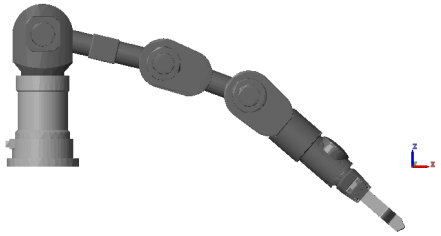


Fig. 13: Manipulator as shown in the Mechanics Explorer.

- 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.

3. 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`). Figure 13 shows the manipulator as displayed in the Mechanics Explorer after importing it with Simulink Simscape.

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 reconstructed the fault signals $\phi(x_1, x_2, u)$, after that, using equation (7) we get the torques exerted on each joint. 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 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 14 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 System 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 System Toolbox. We implemented the SMO and mass estimator using the

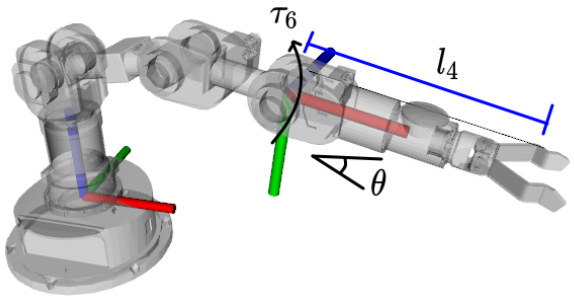


Fig. 14: Katana equivalent variables for mass estimation

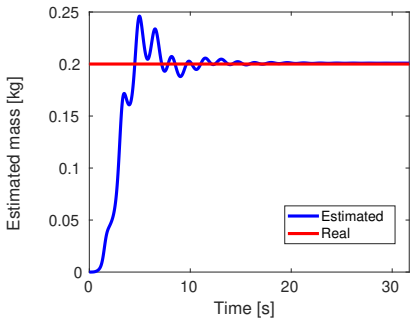


Fig. 15: Mass estimation results with the Katana Manipulator

Forward Dynamics and Mass Matrix block of the Simulink Robotics Toolbox. We simulated an object with a mass of 200 g. To simplify simulations, we did not implement a control but we used only a constant torque applied to the joints. Figure 15 shows the resulting estimation.

All files and instructions necessary to reproduce the results presented in this work can be found in
`https://github.com/RobotJustina/RCF-MathWorks-2020-14/`.

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 System Toolbox, Simscape and ROS Toolbox, 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 use of the implemented 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.

Acknowledgements Authors thank to the MathWorks Students Competitions Team, Control Systems Team and Advanced Research and Technology Office, for their help in the development of this project.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation, pp. 265–283 (2016)
2. Abid, A., Khan, M.T., Iqbal, J.: A review on fault detection and diagnosis techniques: basics and beyond. Artificial Intelligence Review pp. 1–26 (2020)
3. Alwi, H., Edwards, C., Tan, C.P.: Fault detection and fault-tolerant control using sliding modes. Springer Science & Business Media (2011)
4. Billard, A., Kragic, D.: Trends and challenges in robot manipulation. Science **364**(6446), eaat8414 (2019). DOI 10.1126/science.aat8414. URL `https://www.sciencemag.org/lookup/doi/10.1126/science.aat8414`
5. Bohg, J., Morales, A., Asfour, T., Kragic, D.: Data-driven grasp synthesis—a survey. IEEE Transactions on Robotics **30**(2), 289–309 (2013)
6. van der Burgh, M., Lunenburg, J., Appeldoorn, R., van Beek, L., Geijsberts, J., Janssen, L., van Dooren, P., van Rooy, H., Aggarwal, A., Aleksandrov, S., Dang, K., Hofkamp, A., van Dinther, D., van de Molengraft, M.: Tech united eindhoven @home2020 team description paper. https://www.techunited.nl/uploads/Tech_United_At_Home_TDP_2020.pdf (2020)

7. Capisani, L.M., Ferrara, A., De Loza, A.F., Fridman, L.M.: Manipulator fault diagnosis via higher order sliding-mode observers. IEEE Transactions on industrial electronics **59**(10), 3979–3986 (2012)
8. Cordoneanu, D., Niţu, C.: A review of fault diagnosis in mechatronics systems. In: International Conference of Mechatronics and Cyber-Mixmechatronics, pp. 173–184. Springer (2018)
9. Davila, J., Fridman, L., Poznyak, A.: Observation and identification of mechanical systems via second order sliding modes. International Journal of Control **79**(10), 1251–1262 (2006)
10. Di Castro, M., Mulero, D.B., Ferre, M., Masi, A.: A real-time reconfigurable collision avoidance system for robot manipulation. In: Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering, pp. 6–10 (2017)
11. Ding, S.X.: Model-based fault diagnosis techniques: design schemes, algorithms, and tools. Springer Science & Business Media (2013)
12. Ekvall, S., Jensfelt, P., Kragic, D.: Integrating active mobile robot object recognition and SLAM in natural environments. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5792–5797. IEEE (2006). DOI 10.1109/IROS.2006.282389
13. Engelberger, J.F.: Robotics in service. MIT Press (1989). OCLC: 19724093
14. Ferrara, A., Incremona, G.P., Sangiovanni, B.: Sliding mode fault diagnosis with vision in the loop for robot manipulators. In: New Trends in Robot Control, pp. 81–105. Springer (2020)
15. Hamidreza Kasaei, S., Melsen, J., van Beers, F., Steenkist, C., Voncina, K.: The state of lifelong learning in service robots: Current bottlenecks in object perception and manipulation. arXiv e-prints pp. arXiv–2003 (2020)
16. Han, S., Ha, H., Zhao, Y., Lee, J.: Assumed model feedforward sliding mode control for a wheeled mobile robot with 3-dof manipulator systems. Journal of Mechanical Science and Technology **31**(3), 1463–1475 (2017)
17. Huang, J., Ge, W., Cheng, H., Xi, C., Zhu, J., Zhang, F., Shang, W.: Real-time obstacle avoidance in robotic manipulation using imitation learning. In: 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 976–981. IEEE (2020)
18. Jae-Han Park, Seung-Ho Baeg, Jaehan Koh, Kyung-Wook Park, Moon-Hong Baeg: A new object recognition system for service robots in the smart environment. In: 2007 International Conference on Control, Automation and Systems, pp. 1083–1087. IEEE (2007). DOI 10.1109/ICCAS.2007.4407060. URL `http://ieeexplore.ieee.org/document/4407060/`
19. Jin, J.: A robust zeroing neural network for solving dynamic nonlinear equations and its application to kinematic control of mobile manipulator. Complex & Intelligent Systems **7**(1), 87–99 (2021)
20. Jin, L., Li, S., Yu, J., He, J.: Robot manipulator control using neural networks: A survey. Neurocomputing **285**, 23–34 (2018)
21. Khan, A.H., Li, S., Chen, D., Liao, L.: Tracking control of redundant mobile manipulator: An rnn based metaheuristic approach. Neurocomputing **400**, 272–284 (2020)
22. Lee, J., Chang, P.H., Jin, M.: Adaptive integral sliding mode control with time-delay estimation for robot manipulators. IEEE Transactions on Industrial Electronics **64**(8), 6796–6804 (2017)
23. Loncomilla, P., Ruiz-del Solar, J., Martínez, L.: Object recognition using local invariant features for robotic applications: A survey. Pattern Recognition **60**, 499–514 (2016). DOI 10.1016/j.patcog.2016.05.021. URL `https://linkinghub.elsevier.com/retrieve/pii/S0031320316301054`
24. MathWorks-Students-Competitions-Team: Designing robot manipulator algorithms (https://github.com/mathworks-robotics/designing-robot-manipulator-algorithms). GitHub. Retrieved April 6, 2020. (2020)
25. MathWorks-Students-Competitions-Team: Trajectory planning for robot manipulators (https://github.com/mathworks-robotics/trajectory-planning-robot-manipulators). GitHub. Retrieved April 6, 2020. (2020)

26. Memmesheimer, R., Mykhalchyshyna, I., Seib, V., Evers, T., Paulus, D.: homer@UniKoblenz: Winning team of the RoboCup@home open platform league 2018. In: D. Holz, K. Genter, M. Saad, O. von Stryk (eds.) RoboCup 2018: Robot World Cup XXII, vol. 11374, pp. 512–523. Springer International Publishing (2019). DOI 10.1007/978-3-030-27544-0_42. URL http://link.springer.com/10.1007/978-3-030-27544-0_42. Series Title: Lecture Notes in Computer Science
27. Mirrazavi Salehian, S.S., Figueroa, N., Billard, A.: A unified framework for coordinated multi-arm motion planning. The International Journal of Robotics Research **37**(10), 1205–1232 (2018)
28. Nguyen, H., La, H.: Review of deep reinforcement learning for robot manipulation. In: 2019 Third IEEE International Conference on Robotic Computing (IRC), pp. 590–595. IEEE (2019). DOI 10.1109/IRC.2019.00120. URL https://ieeexplore.ieee.org/document/8675643/
29. Nikou, A., Verginis, C., Heshmati-Alamdari, S., Dimarogonas, D.V.: A nonlinear model predictive control scheme for cooperative manipulation with singularity and collision avoidance. In: 2017 25th Mediterranean Conference on Control and Automation (MED), pp. 707–712. IEEE (2017)
30. Omali, K.O., Kabbaj, M.N., Benbrahim, M.: Nonlinear observer-based fault detection and isolation for a manipulator robot. In: New Developments and Advances in Robot Control, pp. 163–185. Springer (2019)
31. Rakita, D., Mutlu, B., Gleicher, M.: Relaxedik: Real-time synthesis of accurate and feasible robot arm motion. In: Robotics: Science and Systems, pp. 26–30. Pittsburgh, PA (2018)
32. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788 (2016)
33. Sahbani, A., El-Khoury, S., Bidaud, P.: An overview of 3d object grasp synthesis algorithms. Robotics and Autonomous Systems **60**(3), 326–336 (2012)
34. Savage, J., LLarena, A., Carrera, G., Cuellar, S., Esparza, D., Minami, Y., Peñuelas, U.: Virbot: a system for the operation of mobile robots. In: RoboCup 2007: Robot Soccer World Cup XI, pp. 512–519. Springer (2008)
35. Seo, I.S., Han, S.I.: Dual closed-loop sliding mode control for a decoupled three-link wheeled mobile manipulator. ISA transactions **80**, 322–335 (2018)
36. Shtessel, Y., Edwards, C., Fridman, L., Levant, A.: Sliding mode control and observation. Springer (2014)
37. Ruiz-del Solar, J., Correa, M., Verschae, R., Bernuy, F., Loncomilla, P., Mascaró, M., Riquelme, R., Smith, F.: Bender – a general-purpose social robot with human-robot interaction abilities. Journal of Human-Robot Interaction **1**(2), 54–75 (2013). DOI 10.5898/JHRI.1.2.Ruiz-del-Solar. URL http://dl.acm.org/citation.cfm?id=3109692
38. Steinbauer, G.: A survey about faults of robots used in robocup. In: Robot Soccer World Cup, pp. 344–355. Springer (2012)
39. Stückler, J., Schwarz, M., Behnke, S.: Mobile manipulation, tool use, and intuitive interaction for cognitive service robot cosero. Frontiers in Robotics and AI **3** (2016). DOI 10.3389/frobt.2016.00058. URL http://journal.frontiersin.org/article/10.3389/frobt.2016.00058/full
40. Sun, Y., Zhang, Z., Leibold, M., Hayat, R., Wollherr, D., Buss, M.: Protective control for robot manipulator by sliding mode based disturbance reconstruction approach. In: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1015–1022. IEEE (2017)
41. Varela-Aldás, J., Andaluz, V.H., Chicaiza, F.A.: Modelling and control of a mobile manipulator for trajectory tracking. In: 2018 International Conference on Information Systems and Computer Science (INCISCOS), pp. 69–74. IEEE (2018)
42. Wachsmuth, S., Holz, D., Rudinac, M., Ruiz-del Solar, J.: Robocup@ home-benchmarking domestic service robots. In: AAAI, pp. 4328–4329 (2015)
43. Wachsmuth, S., Lier, F., Rügemer, L., Meyer zuBorgsen, S.: Tobi - team of bielefelda human-robot interaction system forrobocup@home 2019. https://www.cit-ec.de/sites/default/files/files/tobi/robocupathome-sspl_tobi_tdp-2019.pdf (2019)

44. Wisspeintner, T., van der Zan, T., Iocchi, L., Schiffer, S.: RoboCup@home: Results in benchmarking domestic service robots. In: J. Baltes, M.G. Lagoudakis, T. Naruse, S.S. Ghidary (eds.) RoboCup 2009: Robot Soccer World Cup XIII, vol. 5949, pp. 390–401. Springer Berlin Heidelberg (2010). DOI 10.1007/978-3-642-11876-0_34. URL http://link.springer.com/10.1007/978-3-642-11876-0_34
45. Wisspeintner, T., Zant, T.v.d., Iocchi, L., Schiffer, S.: RoboCup@home: Scientific competition and benchmarking for domestic service robots. Interaction Studies. Social Behaviour and Communication in Biological and Artificial Systems **10**(3), 392–426 (2009). DOI 10.1075/is.10.3.06wis. URL http://www.jbe-platform.com/content/journals/10.1075/is.10.3.06wis
46. Yamada, J., Lee, Y., Salhotra, G., Pertsch, K., Pflueger, M., Sukhatme, G.S., Lim, J.J., Englert, P.: Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. arXiv preprint arXiv:2010.11940 (2020)
47. Zhang, S., Li, Y., Liu, S., Shi, X., Chai, H., Cui, Y.: A review on fault-tolerant control for robots. In: 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 423–427. IEEE (2020)