Model predictive control for assistive robotics manipulation

Fábio Luz[1], Bruno Guerreiro[2,1], and Manuel Marques[1]

[1] ISR/LARSYS, Instituto Superior Técnico, Univ. Lisboa, Lisboa, Portugal
[2] DEEC and CTS, NOVA School of Science and Technology, Caparica, Portugal
E-mails: fabiodpluz@tecnico.ulisboa.pt, bj.guerreiro@fct.unl.pt, manuel@isr.tecnico.ulisboa.pt

Abstract. This paper addresses the use of Model Predictive Control (MPC) method for robotic manipulation, more specifically for a six-degrees of freedom robotic manipulator that is used for assistive purposes. MPC does not require any user's control input, allows the handle of nonlinear constraints, and opens the door to integrate machine learning and artificial intelligence techniques, contributing towards more autonomous assistance. The results show the ability to reach the reference position, even if it is variable through time, and to react immediately to perturbations, adapting the trajectory of the end-effector.

1 Introduction

In fundamental human value activities' like eating a meal, the use of a joystick to control the manipulator [1] or the use of input buttons to select the type of food in the plate that is going to travel to a predefined mouth's position, as featured by the Handy 1 robot [2], are a few examples of Assistive Robotic Manipulators (ARMs) that can be controlled by the users [3, 4]. However, it requires training to get familiarized with the system, and some users can not control the inputs due to their disabilities.

In recent years, many studies have been made on manipulators and their use to give assistance to humans in ADL like eating [2], [5], [6], [7], dressing [8], and drinking [9]. One of the first low-cost robots to be developed and commercially available to aid severely disabled people was M. Topping's Handy 1 [2], yet it used a predefined position for the users' mouth.

To surpass this problem, in [5] the system has a man-machine interface controlled by the individual's head-motion, [6] offers an eye-tracking technology to control the manipulator, and a voice-controlled system is proposed in [8] for dressing. As these manipulators lack adaptation to the user in real-time, other work focused on using Brain-Machine Interfaces (BMIs) to assist in the task of drinking [9].

In project FEEDBOT [7], an RGB-D camera was used to detect the environment and track the users' mouth movement. It adapts the trajectory in real-time to the users' movements by using Discriminative Optimization [10] for tracking

the users' head, rather than having a pre-programmed mouth's position. Using this solution it is possible to design fully autonomous robots that can perceive the environment and act accordingly.

Towards this goal, an important step is to design advanced controllers used to fulfill the tasks attributed to the manipulators or even integrate in the control algorithm data-driven techniques. Standard control techniques such as PID control, may not be able to cope with physical and environmental constraints to the ARM operation to make it as autonomous as possible.

As such the goal of this paper is to design an algorithm based on model predictive control (MPC) for a six-DOF robotic arm used for aid in ADL like assistive feeding purposes. Given a target position of the end-effector, the controller must be able to plan a trajectory and adapt in real-time to perturbations on its conditions, such as disturbances on the position of the end-effector and changes on its final reference position, the users' mouth.

MPC uses a dynamic model of a system to estimate its behavior and determine the best control options [11]. In [12] and [13], the authors study the use of nonlinear MPC (NMPC) with different parameters in industrial robots for path-following using predetermined trajectories. It is also possible to devise strategies that combines MPC with vision and deep neural networks to achieve fast and safe tracking control for a robotic manipulators, satisfying constraints and stability requirements [14, 15]. The present work follows this line of research, with a first step of applying NMPC to a concrete ARM developed under Project FEEDBOT [7].

The contributions of this paper include the development of the ARM forward kinematics and a simple dynamic model of the ARM based on obtained experimental data from a six-DOF Niryo One [16], as well as an MPC technique for trajectory planning and control of the ARM given a final desired position of the end-effector. Simulation results are also provided to validate the proposed technique.

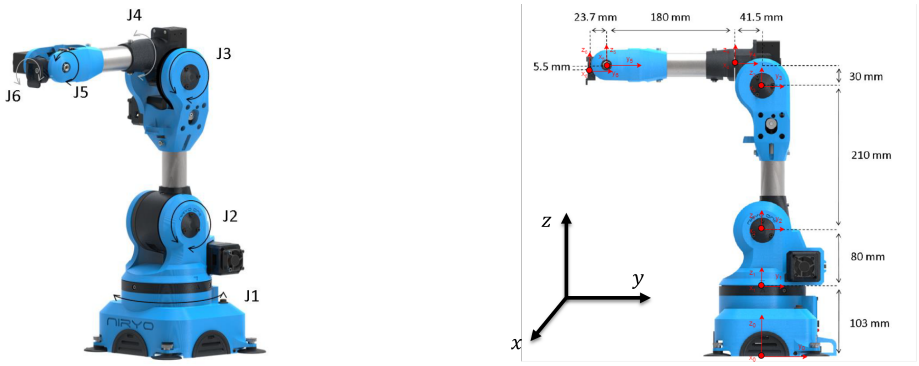
The remainder of this paper is organized as follows. Section 2 introduces de ARM forward kinematics and dynamic models, while Section 3 presents the proposed MPC strategy to control the ARM. The simulation results are presented in Section 4, whereas some concluding remarks and future work are provided in Section 5.

2 Six-DOF robotic arm

The manipulator used in this work is the Niryo One robot arm. It is considered a serial manipulator or a kinematic chain since its design is composed of several links or rigid bodies connected by joints.

As presented in Figure (1a), Niryo One is a six-DOF robotic arm. Each link can only move in one direction, and so each DOF corresponds to a joint. It has three twisting joints and three rotational joints:twisting joints are the ones where the axis of rotation is parallel to the axis of the two links, rotating in a twisting motion (J1, J4, and J6), while rotational joints are the ones where the two links

rotate relative to each other about a fixed axis, perpendicular to the links (J2, J3, and J5).



(a) 6 DOF joints.

(b) Dimensions and coordinate frames

Fig. 1: Niryo One robotic manipulator.

In the center of each joint is defined a coordinate frame to be possible to compute the position of the end-effector relative to the base frame from the angles of the joints. The frame system for each joint is presented in Figure 1b, where for each joint the x-axis pointing towards the reader, the y-axis pointing to the right, and with the z-axis pointing up for the presented configuration. The angular limitations of each joint and the physical dimensions or each link are also detailed in Table 1.

Table 1: Limits of each joint and dimensions of each link

Joint	**Min (rad)**	**Max (rad)**
J1	-3.054	3.054
J2	-1.5707	0.628319
J3	-1.4101	0.994838
J4	-2.61799	2.61799
J5	-2.26893	2.26893
J6	-2.57	2.57

Link	**Dim. (mm)**
L_1	103
L_2	80
L_3	210
L_4	30
L_5	41.5
L_6	180
L_7	23.7
L_8	5.5

2.1 Forward kinematics

The direct kinematics consists in finding the position and orientation of the end-effector, which is located in J6, relative to the base referential defined in the

base of the serial manipulator, given the value of the angles of each joint, θ_i for $i = 1, ..., 6$.

Knowing the angle of the joint between two consecutive links, it is possible to compute the homogeneous transformation between them. Let R_i^j be the rotation matrix that transforms a vector in the i-th coordinate frame to a coordinate frame which is parallel to the j-th coordinate frame and p_i^j be the position of the i-th coordinate frame with respect to the j-th coordinate frame. The transformation matrix T_i^j is a concatenation of the 3-by-3 rotation matrix with the 3-by-1 position array, both between two consecutive coordinate frames and describes the frame i relative to the frame j, which can be represented as

$$T_i^j = \begin{bmatrix} R_i^j & p_i^j \\ 0_{3\times1} & 1 \end{bmatrix} \tag{1}$$

Considering the coordinate system defined above, the transformation matrix between the base referential and the frame in joint J1 is given by

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where $c_i = \cos(\theta_i)$ and $s_i = \sin(\theta_i)$. For completeness, the following transformations between joints can be observed:

- Base $\rightarrow$ J1: rotation about z-axis, translation of L_1 in z-axis, as in Eq. (2).
- J1 $\rightarrow$ J2: rotation about x, translation of L_2 in z.
- J2 $\rightarrow$ J3: rotation about x, translation of L_3 in z.
- J3 $\rightarrow$ J4: rotation about y, translation of $-L_5$ in y and L_4 in z.
- J4 $\rightarrow$ J5: rotation about x, translation of $-L_6$ in y.
- J5 $\rightarrow$ J6: rotation about y, translation of $-L_7$ in y and $-L_8$ in z.

With all the transformation defined between the consecutive frames, it is possible to compute the relation between the base referential frame and the coordinate frame of J6 where the end-effector is located as

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 = \begin{bmatrix} R_6^0 & p_6^0 \\ 0_{3\times1} & 1 \end{bmatrix} \tag{3}$$

2.2 Model dynamics

A dynamic model is required to simulate with accuracy the behavior of the robotic arm and contemplates not only the torques applied by the motor actuators present in each joint but also the external forces suffered by the manipulator, and its general expression is described in [17] by

$$\tau = M(x)\ddot{x} + C(x, \dot{x}) + G(x) \tag{4}$$

where τ represents the actuators' torques while x, $\dot{x}$ and $\ddot{x}$ are the position, velocity, and acceleration of the joints, respectively. $M(x)$ denotes the inertia or mass matrix, $C(x, \dot{x})$ is the vector that includes the Coriolis and centrifugal forces, and $G(x)$ is the gravity vector.

Given that the maker of this arm does not provide the necessary information, the procedure to identify the center of mass and moment of inertia of each link is a complex task, which requires disassembling the manipulator and using specialized measuring apparatus. On the other hand, the type of actuators used in each joint (stepper motors) takes as a command a desired angular position and not a current or torque.

Consequently, a much simpler system model was defined and used in the simulation that only considers the angular position's dynamics of the joints and admits a control input to be a desired angular position, defining

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= p_6^0(x(t)) \end{aligned} \tag{5}$$

where x is the vector of joint angles, $x = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}^T$, and the control signal contains the desired joint angles, $u = \begin{bmatrix} \theta_{1d} & \theta_{2d} & \theta_{3d} & \theta_{4d} & \theta_{5d} & \theta_{6d} \end{bmatrix}^T$. The system output is defined by the position of the end-effector, located on J6, in the base referential frame, p_6^0. Note that this computation is the result of applying the six successive joints transformation as defined in (3).

For simplicity, the system and input matrices are assumed diagonal, where $A = \text{diag}(a_1, a_2, a_3, a_4, a_5, a_6)$ and $B = diag(b_1, b_2, b_3, b_4, b_5, b_6)$. Considering that the actuation at each joint is achieved using a stepper motor, which has a static error close to zero in regular operational conditions, we consider a further simplification by assuming that $b_i = -a_i$. This is equivalent to assume a zero static gain in a first order linear model.

To identify the system, the Niryo One's *Matlab* interface [18] was used to send simple commands to the manipulator joints to analyze its response in real-time. When a set of angles is sent to control the joints, a trajectory is created and passed to the interface controller that executes it, as can be seen in Figure 2 for joint J3. Although it can be clearly seen that this is not the response of a first order linear system, to simplify the resulting model, we assume that the obtained response is the result of a step in the desired joint angle, and proceed to identify a_i from the time the experimental response takes to reach 63.2% of its final value.

From a comprehensive set of experimental tests for all joints, we concluded that all joints have a similar response to the same command, with a similar time constant of approximately 1.2 seconds. As such, it is considered that for all the joints the model parameters are $a_i = -0.83$ and $b_i = 0.83$.

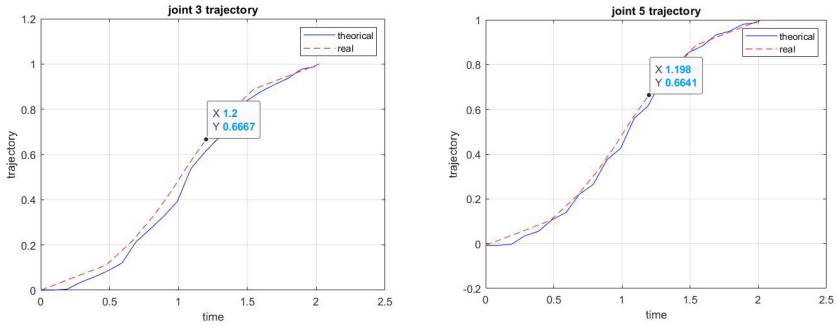


Fig. 2: Joints 3 and 5 responses to a change of 1 radian

3 Model Predictive Control

Model Predictive Control (MPC) is a control algorithm that uses an explicit dynamic model in order to make predictions about the future behavior of a given system in response to the actuation of the manipulated variables.

Based on the state predictions and having in consideration the reference or desired state trajectory, an optimal control problem (OCP) is solved to select the optimal control action subject to operating constraints [19]. The generic optimization problem solved by MPC to determine that sequence, given an initial state, x_0, can be defined in continuous-time as

$$\min_{x,u} \quad m(x(T)) + \int_0^T l(x(\tau), u(\tau))d\tau \tag{6}$$

$$\text{s. t.} \quad \dot{x} = f(x(t), u(t)) \qquad t \in [0, T] \tag{7}$$

$$\underline{h} \leq h(u(t)) \leq \overline{h} \qquad t \in [0, T] \tag{8}$$

$$\underline{x} \leq x(t) \leq \overline{x} \qquad t \in [0, T) \tag{9}$$

$$\underline{x}^e \leq x(T) \leq \overline{x}^e \tag{10}$$

$$x(0) = x_0 \tag{11}$$

where T is the prediction horizon, $m(.)$ and $l(.)$ are the Mayer and Lagrange cost functions to be specified, constraint (7) represents the possibly nonlinear model of the system, and the remaining constraints specify the input, state, and terminal state boundaries, as well as the initial state condition.

The MPC simulates multiple actuation scenarios in the manipulated variables and the respective model response and chooses the control sequence that minimizes the defined cost function, that usually includes the difference between the desired trajectory and the predicted one and the variation in input control signal between consecutive iterations, but that also respects the constraints. After the optimal control sequence is determined, the first value is applied in the system. In the next sampling time a new state measurement can be obtained and the optimization procedure is then repeated.

3.1 Implementation of MPC for a six-DOF ARM

The simulation algorithm was developed using the *MATLAB* interface of *ACADOS*, a software package for the efficient solution of optimal control and estimation problems [20] that also makes use of *CasADi*, an open-source tool for nonlinear optimization and algorithmic differentiation [21]. The simulation algorithm diagram is represented in Figure 3.

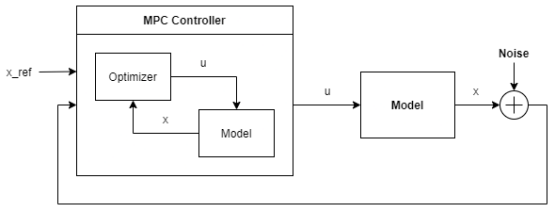


Fig. 3: Simulation diagram

To initialize the MPC, the initial state of the joints, x_0, and the reference position at end-time, y^e_{ref}, are generated according to the purpose of the simulation (fixed or variable reference position, physical limitations). The dynamics (7), are formulated as explicit equations in continuous time, as written in (5).

In addition the initial state, x_0, it is necessary to have in consideration the physical limitations of the arm both in the state and in the input since the control is the desired position of the joints. Those limitations are stated in Table 1, used to define $\underline{h} = \underline{x} = \underline{x}^e$ and $\overline{h} = \overline{x} = \overline{x}^e$. The Lagrangian cost term, $l(x, u)$, and the Mayer cost term, $m(x)$ are given by

$$l(x,u) = \frac{1}{2}\|y(x,u) - y_{ref}\|^2_W \tag{12}$$

$$m(x) = \frac{1}{2}\|y^e(x) - y^e_{ref}\|^2_{W^e} \tag{13}$$

where W and W^e represent weight matrices that can be tuned, and $y(x, u)$ and $y^e(x)$ are defined as

$$y(x,u) = [p^0_6(x), \dot{x}(x,u)] \tag{14}$$

$$y^e(x) = p^0_6(x) \tag{15}$$

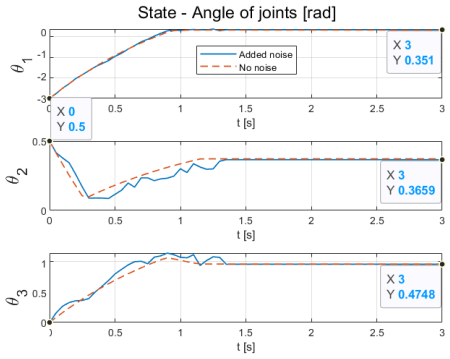
The Lagrangian term not only has into consideration the position of the end-effector of the robotic arm, $p^0_6(x)$, but also the velocity of the joints, $\dot{x}(x, u)$, to improve stability and decrease rash movements. The Mayer term is only added at the end of the optimization process, and it can not depend on the control signal, u, so it is only defined by the position of the end-effector.

4 Simulation results

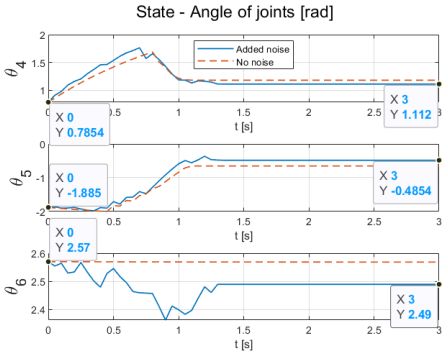
In this section several simulation results are presented to validate the proposed MPC methodology. All the results presented in this section consider a prediction horizon of $T = 0.5$ seconds and 10 multiple shooting nodes. To have more realistic simulations, random noise is added to the angular position of each joint, ranging from -0.05 and 0.05 rad.

Three different scenarios are considered: (a) a pair of an initial state and a final reference position for the end-effector of the robotic arm, (b) two different reference positions in the same trajectory, and (c) a reference position that is moving circularly through the simulation. For all these scenarios the state and output evolution are shown, that is, the angle of each joint and the position of the end-effector during the simulation. In addition, the solid blue line represents the simulation with added noise, and the black dotted line and cross-markers represent the reference position of the end-effector.

The results for scenario (a) are presented in Figure 4, where a valid initial state and a reachable reference position are chosen such that an angle of $\pi/8$ radians for all joints is considered. There are several combinations of joint angles



(a) Angle of J1 to J3.



(b) Angles of J4 to J6.

Fig. 4: Scenario (a): joint angles.

that lead to the identical position of the end-effector, and it is possible to observe in Figure 4 that the MPC algorithm reaches the same position with different angles, depending if noise is considered or not.

For the particular case of this project, the possibility of changing the reference position during the trajectory of the end-effector, and not having a fixed one is very important to meet the objective of applying this control technique to a robotic arm used for assistive feeding purposes, since the mouth of the person being fed (the reference position) is movable in the world space.

In simulation scenario (b), two different reference positions alternate every two seconds of the simulation, as presented i Figure 5. For a better understanding of the results, the simulation time was increased to ten seconds.

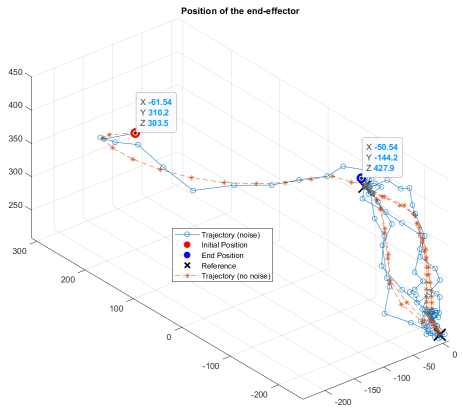
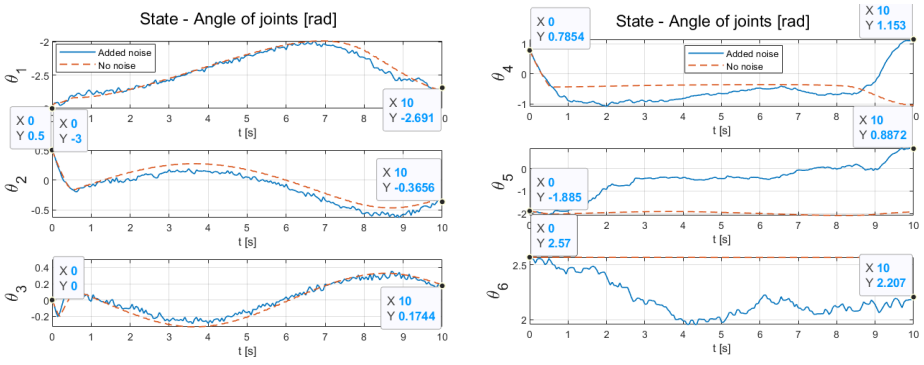


Fig. 5: Position of the end-effector (J6) - scenario 2

As can be observed, the reference position is followed by the model of the robotic arm, and the algorithm adapts quite quickly to the change in the reference position without making abrupt variations in the movement of the end-effector. The trajectory of the end-effector when the noise is added is very similar to when it is not.

The results for scenario (c) in which the reference position changes through time are presented in Figures 6 and 7, where the reference trajectory of the end-effector describes a circumference. It can be seen that the end-effector follows



(a) Angle of J1 to J3.

(b) Angles of J4 to J6.

Fig. 6: Scenario (c): joint angles.

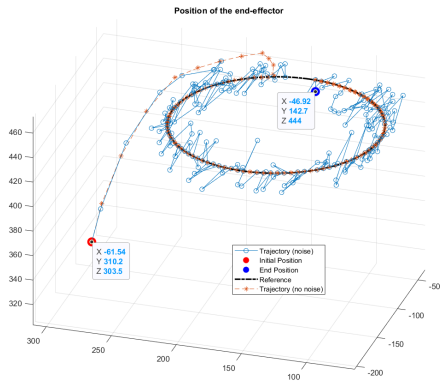


Fig. 7: Position of the end-effector (J6) - scenario 3

the circular reference, although the precise position is not met because the MPC does not have time to settle once the reference changes in every iteration given the addition of noise in the current position of the arm. For the simulation without added noise, once the end-effector reaches a desired reference it follows its trajectory flawlessly.

5 Conclusions

In this work, the structure of a six-DOF robotic manipulator was studied to the extent of reaching the main objective that was to design an algorithm based on the MPC method that solves Optimal Control Problems (OCP) to plan the trajectory for the end-effector to a fixed or variable reference, adapting in real-time to disturbances in its position.

The geometrical properties of the manipulator were analyzed to compute the forward kinematics, the correspondence between the angles of the six joints of the arm, and the position of the end-effector in a base reference frame. Since it was not possible to establish a complete dynamic model for the robotic arm, a a simpler differential system's model was defined to represent it in the simulation, and its parameters were estimated from the response to simple commands.

Based on the simulation results, it is confirmed that the proposed MPC strategy can follow a reference position, whether it is fixed or variable through time, reacting to the perturbations imposed by the addition of noise to the end-effector in each iteration of the algorithm while respecting the physical constraints of the manipulator and while generating and maintaining a relatively steady trajectory. It is possible to observe that when no noise is added to the position of the end-effector, the trajectory is cleaner, but its presence does not significantly affect the smoothness of the joints' movement nor the performance of the algorithm.

Future work will focus on the experimental validation of the proposed algorithm using the Niryo One robotic arm. With these experiments, it will be

possible to verify if the simplified model used in the simulation is enough to satisfy the requirements of the feeding task or if a full model of the arm is necessary to improve the results. Also, the parameters' identification using more advanced techniques might be helpful when applying the algorithm to the concrete ARM to achieve satisfactory trajectories.

Once the MPC is functioning successfully in reality, vision and machine learning techniques, such as deep and reinforcement learning, can be used to improve the algorithm by integrating with the optimization the visual information of the environment, like the individuals' expressions while eating.

Acknowledgments

This work was partially funded by FCT project REPLACE (PTDC/EEIAUT-/32107/2017) which includes Lisboa 2020 and PIDDAC funds, project CAPTURE (PTDC/EEI-AUT/1732/2020), and also projects CTS (UIDB/00066-/2020) and LARSYS (UIDB/50009/2020).

References

1. Reem K. Al-Halimi and Medhat Moussa. Performing complex tasks by users with upper-extremity disabilities using a 6-dof robotic arm: A study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(6):686–693, 2017.
2. Mike Topping. An overview of the development of handy 1, a rehabilitation robot to assist the severely disabled. *Journal of Intelligent and Robotic Systems*, 34:253–263, 07 2002.
3. D. Feil-Seifer and M.J. Mataric. Defining socially assistive robotics. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 465–468, 2005.
4. Anton Satria Prabuwono, Khalid Allehaibi, and Kurnianingsih Kurnianingsih. Assistive robotic technology: A review. *Computer Engineering and Applications Journal*, 6:71–78, 07 2017.
5. S. Ishii, S. Tanaka, and F. Hiramatsu. Meal assistance robot for severely handicapped people. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 2, pages 1308–1313 vol.2, 1995.
6. P. Lopes, R. Lavoie, R. Faldu, N. Aquino, J. Barron, M. Kante, B. Magfory, and W. Meleis. Eye-controlled robotic feeding arm technology (icraft). page 1–24, 2012.
7. Alexandre Candeias, Travers Rhodes, Manuel Marques, João P. Costeira, and Manuela Veloso. Vision augmented robot feeding. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 50–65. Springer International Publishing, 2019.
8. Greg Chance, Antonella Camilleri, Benjamin Winstone, Praminda Caleb-Solly, and Sanja Dogramadzi. An assistive robot to support dressing - strategies for planning and error handling. *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 774–780, 06 2016.
9. S. Schröer, Ingo Killmann, Barbara Frank, Martin Völker, Lukas Fiederer, Tonio Ball, and Wolfram Burgard. An autonomous robotic assistant for drinking. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015:6482–6487, 06 2015.

10. J. Vongkulbhisal, F. De la Torre, and J. Costeira. Discriminative optimization: Theory and applications to computer vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 41(04):829–843, apr 2019.
11. J Rawlings and D.Q. Mayne. *Model Predictive Control: Theory, Computation and Design*. Nob Hill Publishing, LLC, 2nd edition, 01 2009.
12. Timm Faulwasser, Tobias Weber, Pablo Zometa, and Rolf Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2017.
13. Niels van Duijkeren, Robin Verschueren, Goele Pipeleers, Moritz Diehl, and Jan Swevers. Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation. In *2016 European Control Conference (ECC)*, pages 477–482, 2016.
14. Julian Nubert, Johannes Köhler, Vincent Berenz, Frank Allgöwer, and Sebastian Trimpe. Safe and fast tracking control on a robot manipulator: Robust MPC and neural network control. *CoRR*, abs/1912.10360, 2019.
15. Paul Drews, Grady Williams, Brian Goldfain, Evangelos Theodorou, and James Rehg. Aggressive deep driving: Model predictive control with a cnn cost model. 07 2017.
16. Niryo One - Description and mechanical specifications (Website). Accessed 10-July-2021.
17. Li Ding, Hongtao Wu, Yu Yao, and Yuxuan Yang. Dynamic model identification for 6-dof industrial robots. *Journal of Robotics*, 2015:1–9, 10 2015.
18. Niryo One - MATLAB interface (Website). Accessed 10-July-2021.
19. Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.
20. Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados: a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 2021.
21. Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.