Machine Learning-based Framework for Optimally Solving the Analytical Inverse Kinematics for Redundant Manipulators

M.N. Vu[a,*], F. Beck[a], M. Schwegel[a], C. Hartl-Nesic[a], A. Nguyen[b], A. Kugi[a,c]

[a]*Automation & Control Institute (ACIN), TU Wien, Vienna, Austria*
[b]*Department of Computer Science, University of Liverpool, Liverpool, England*
[c]*Center for Vision, Automation & Control, Austrian Institute of Technology GmbH (AIT), Vienna, Austria*

Abstract

Solving the analytical inverse kinematics (IK) of redundant manipulators in real time is a difficult problem in robotics since its solution for a given target pose is not unique. Moreover, choosing the optimal IK solution with respect to application-specific demands helps to improve the robustness and to increase the success rate when driving the manipulator from its current configuration towards a desired pose. This is necessary, especially in high-dynamic tasks like catching objects in mid-flights. To compute a suitable target configuration in the joint space for a given target pose in the trajectory planning context, various factors such as travel time or manipulability must be considered. However, these factors increase the complexity of the overall problem which impedes real-time implementation. In this paper, a real-time framework to compute the analytical inverse kinematics of a redundant robot is presented. To this end, the analytical IK of the redundant manipulator is parameterized by so-called redundancy parameters, which are combined with a target pose to yield a unique IK solution. Most existing works in the literature either try to approximate the direct mapping from the desired pose of the manipulator to the solution of the IK or cluster the entire workspace to find IK solutions. In contrast, the proposed framework directly learns these redundancy parameters by using a neural network (NN) that provides the optimal IK solution with respect to the manipulability and the closeness to the current robot configuration. Monte Carlo simulations show the effectiveness of the proposed approach which is accurate and real-time capable ($\approx 32\,\mu s$) on the KUKA LBR iiwa 14 R820.

Keywords: redundant manipulator, analytical inverse kinematics; numerical inverse kinematics; machine learning; trajectory optimization.

1. Introduction

The inverse kinematics (IK) [39] solution is fundamental for many applications in robotics involving motion planning, e.g., point-to-point trajectory optimization [44, 20], path-wise trajectory planning [18, 11], dexterous grasping [14, 32], and pick-and-place scenarios [35, 30]. Solving the IK problem for a given target position in the task space yields the robot's configuration in joint space that satisfies the kinematic constraints [26].

There are three types of techniques to solve IK problems, i.e. the algebraic approach, see, e.g., [33, 6], the analytical (or so-called geometric) approach, see, e.g., [37, 45], and the numerical (or so-called iterative) approach, see, e.g., [34, 38]. In the algebraic approach, essentially systems of polynomial equations [29] are solved. Typically, they are classified as difficult algebraic computational problems [6]. In general, this algebraic problem can be solved for a manipulator with 6 degrees of freedom (DoF), see, e.g., [7], but is not generally applicable to kinematically redundant manipulators [36]. On the other hand, numerical methods, typically based on least-squares or pseudoinverse formulations, are widely employed, see, e.g., [46], for various kinematic structures due to their simplicity, low computing time, and their generality. However, these methods may converge to a local minimum that predominantly depends on the initial guess of the solution.

In contrast to the numerical IK, the analytical IK computes the exact solution, which is important for many industrial applications [48, 12]. The computing time of the analytical IK solutions is much faster and real-time capable, compared to the numerical approach. While the analytical IK is only available for specific robot kinematics, most industrial robots are designed such that an analytical solution of the IK is available. Hence, the IK of 6-DoF industrial robots with a spherical wrist, non-offset 7 DoF S-R-S redundant manipulators [41, 21], e.g., KUKA LBR iiwa 14 R820, but also the offset redundant manipulator, e.g., Franka Emika Panda [10], OB7 [31], and Neura Robotics LARA [28], can be solved analytically. These manipulators are often referred to as collaborative robots (Cobot). Typically, the analytical IK parameterizes the robot redundancy by additional (three) parameters, which are usually named redundancy parameters. Examples are [37] and [16] for the non-

*Corresponding author
Email addresses: vu@acin.tuwien.ac.at (M.N. Vu), beck@acin.tuwien.ac.at (F. Beck), schwegel@acin.tuwien.ac.at (M. Schwegel), hartl@acin.tuwien.ac.at (C. Hartl-Nesic), anh.nguyen@liverpool.ac.uk (A. Nguyen), kugi@acin.tuwien.ac.at (A. Kugi)

offset and offset redundant manipulator, respectively. Due to the redundant nature of these parameters, infinite sets of parameters exist in general yielding different IK solutions. However, finding the set of redundancy parameters which represents the *best* IK solution of a specific task is a non-trivial problem.

In this work, a learning-based framework to compute the optimal set of redundancy parameters for an analytical IK is proposed. This improves the computing performance for solving the IK problem, which is essential for highly dynamic tasks like catching objects in mid-flight or handing over objects between moving agents. These tasks are frequently solved using trajectory optimization where typically dynamic system constraints, state and control input constraints, and a target pose constraint are considered.

The target pose constraint is often formulated in the task space and for kinematically redundant robots, in which infinite joint configurations satisfy this constraint. Utilizing the analytical IK in the trajectory optimization problem allows to find the best target joint configuration for a specific task. On the other hand, computing time becomes an issue with this approach. Recently, [45] proposed a real-time capable closed-form solution for the KUKA iiwa 14 R820, where the authors minimize the joint velocities and accelerations while avoiding joint boundaries for a trajectory tracking task. This approach does neither consider the dynamic constraints nor the system state and input constraints in the trajectory optimization.

In addition, it is crucial that the target configuration is well chosen among the infinite solutions of the IK, i.e. close to the initial robot configuration and with high manipulability. For example in a dynamic handover of an object where the target is moving, see Fig. 1, it is advantageous to choose a target configuration that maximizes manipulability such that the robot end-effector can move to another target configuration with high agility. Moreover, choosing a target configuration that is close to the initial configuration of the robot can lead to high performance of the trajectory optimization with a high success rate. Including such criteria in the trajectory optimization is the motivation for the work in this paper.

To this end, a learning-based framework to include additional, application-specific criteria in the analytical IK of redundant robots is proposed. First, a database of 10^8 random pairs of initial configurations and target poses is generated. For each pair, the optimal trajectory is computed using classical approaches, considering application-specific criteria, and is stored in the database together with the set of optimal redundancy parameters. This database serves as the basis to train a neural network (NN), which is used to predict the optimal redundancy parameters for the analytical IK in highly dynamic real-time applications.

The main contribution of this work is a learning-based framework that employs a NN to predict the redundancy parameters of an analytical IK. This yields an optimal target joint configuration for a given target pose by considering application-specific criteria. In this work, the target joint configuration is chosen close to the current joint configuration of the robot and to have a high measure of manipulability. The proposed learning framework significantly speeds up the computing time of the trajectory optimization problem. Note that the proposed framework is tailored to the non-offset redundant manipulator KUKA LBR iiwa 14 R820. Nevertheless, it is also applicable to other kinematically redundant robots with an analytical IK, e.g., [10, 31, 28].

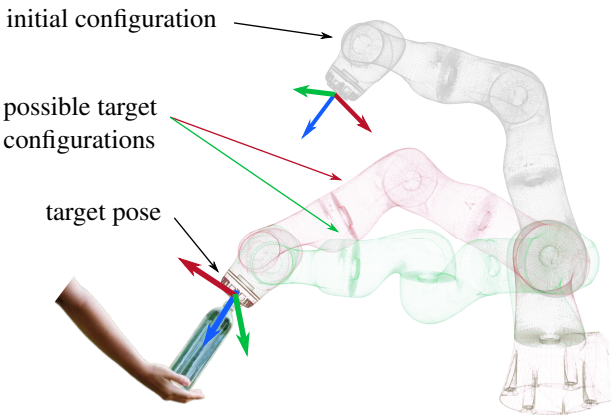

Figure 1: An example of a handover task between robot and human.

The remainder of this paper is organized as follows. Section 2 presents the mathematical modeling and analytical inverse kinematics. Additionally, details of the point-to-point (PTP) trajectory optimization problem and the algorithm for determining the optimal target joint configuration w.r.t. application-specific criteria are given. The learning framework for predicting the redundancy parameters for the analytical IK problem including database generation and the proposed NN is presented in Section 3. Simulation results are shown in Section 4. The last section, Section 5, concludes this work.

2. Trajectory Optimization Framework

This section presents the trajectory optimization framework which is commonly used in robotics [1]. For example, in Figure 1, with a given target pose and a robot's initial configuration, an optimal trajectory in joint space is planned for the robot to catch an object.

In this section, the mathematical modeling of the KUKA LBR iiwa 14 R820, including kinematics and system dynamics, is briefly summarized. Then, the analytical inverse kinematics with the redundancy parameters of this redundant manipulator is presented in Section 2.2. Subsequently, a point-to-point trajectory optimization is performed, which is used to plan trajectories to the optimal target configuration, explained in Section 2.4.

2.1. Mathematical modeling

The KUKA LBR iiwa 14 R820 is an anthropomorphic manipulator due to its similarity to a human arm, which has an S-R-S kinematic structure [41]. The coordinate frames O_i and the corresponding seven revolute joints q_i, $i = 1, \dots, 7$, of the robot

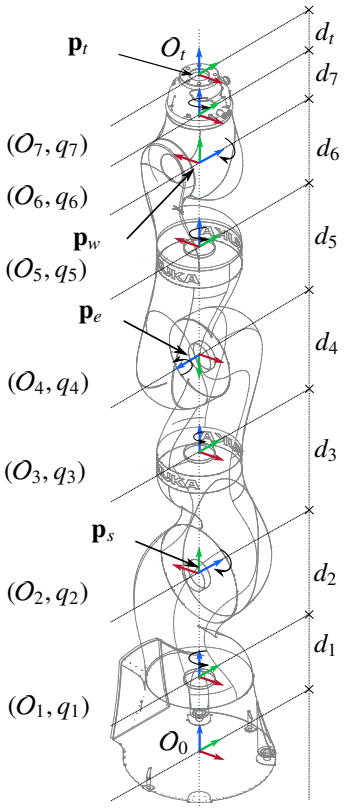

Figure 2: Schematic drawing of the robot KUKA LBR iiwa. The x-, y-, and z-axis of each coordinate frame are shown as red, green, and blue arrows, respectively.

are shown in Fig. 2. The red, green, and blue arrows represent the x-, y-, and z-axis, respectively. The shoulder intersection position $\mathbf{p}_s$ of the joint axes q_1, q_2, and q_3 and the wrist intersection position $\mathbf{p}_w$ of the joint axes q_5, q_6, and q_7 correspond to the shoulder and wrist positions of the human arm, respectively. The elbow position $\mathbf{p}_e$ is in the center of the joint axis q_4.

The robot is modeled as a rigid-body system with the generalized coordinates $\mathbf{q}^{\mathrm{T}} = [q_1, q_2, \ldots, q_7]$, see Fig. 2, which are the rotation angles q_i about the z-axes (blue arrows) of each coordinate frame O_i, $i = 1, \ldots, 7$. To describe the kinematic relationship between the joint angles and the pose of the robot links comprising position and orientation, the homogeneous transformations $\mathbf{T}_n^m$ between two adjacent frames O_n and O_m are constructed, see Tab. 1. Here and in the following, the homogeneous transformation of a simple translation by distance d along the local axis $j \in \{x, y, z\}$ is denoted by $\mathbf{T}_{\mathbf{D},j}(d)$, while an elementary rotation around the local axis $j \in \{x, y, z\}$ by the angle ϕ is described by $\mathbf{T}_{\mathbf{R},j}(\phi)$. The end-effector transformation matrix $\mathbf{T}_0^e$ is referred to as forward kinematics and is computed in the form

$$\mathbf{T}_0^e = \mathrm{FK}(\mathbf{q}) = \prod_{i=0}^{7} \mathbf{T}_i^{i+1} = \begin{bmatrix} \mathbf{R}_0^7(\mathbf{q}) & \mathbf{p}_t(\mathbf{q}) \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{1}$$

comprising the 3D tip position $\mathbf{p}_t \in \mathbb{R}^3$ and the 3D orientation of the end effector as rotation matrix $\mathbf{R}_0^7 \in \mathbb{R}^{3\times3}$. The equations of motion are derived using the Lagrange formalism, see, e.g.,

Table 1: Coordinate transformation of the robot

Frame O_n	Frame O_m	Transformation matrix $\mathbf{T}_n^m$
0	1	$\mathbf{T}_{\mathbf{D},z}(d_1)\mathbf{T}_{\mathbf{R},z}(q_1)$
1	2	$\mathbf{T}_{\mathbf{D},z}(d_2)\mathbf{T}_{\mathbf{R},z}(-\pi)\mathbf{T}_{\mathbf{R},x}(\pi/2)\mathbf{T}_{\mathbf{R},z}(q_2)$
2	3	$\mathbf{T}_{\mathbf{D},y}(d_3)\mathbf{T}_{\mathbf{R},z}(\pi)\mathbf{T}_{\mathbf{R},x}(\pi/2)\mathbf{R}_z(q_3)$
3	4	$\mathbf{T}_{\mathbf{T},z}(d_4)\mathbf{T}_{\mathbf{R},x}(\pi/2)\mathbf{T}_{\mathbf{R},z}(q_4)$
4	5	$\mathbf{T}_{\mathbf{D},y}(d_5)\mathbf{T}_{\mathbf{R},z}(\pi)\mathbf{T}_{\mathbf{R},x}(\pi/2)\mathbf{T}_{\mathbf{R},z}(q_5)$
5	6	$\mathbf{T}_{\mathbf{D},y}(d_6)\mathbf{T}_{\mathbf{R},x}(\pi/2)\mathbf{T}_{\mathbf{R},z}(q_6)$
6	7	$\mathbf{T}_{\mathbf{D},z}(d_7)\mathbf{T}_{\mathbf{R},z}(\pi)\mathbf{T}_{\mathbf{R},x}(\pi/2)\mathbf{T}_{\mathbf{R},z}(q_7)$
7	t	$\mathbf{T}_{\mathbf{D},y}(d_t)$

Table 2: Kinematic and dynamic limits of the system

Joint i	Joint limits $\overline{q_i}$ (°)	Velocity limits $\overline{\dot{q}_i}$ (°/s)	Torque limits $\overline{\tau_i}$ (N)
1	170	85	320
2	120	85	320
3	170	100	176
4	120	75	176
5	170	130	110
6	120	135	40
7	175	135	40

[39],

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{2}$$

where $\mathbf{M}(\mathbf{q})$ denotes the symmetric and positive definite mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis matrix, $\mathbf{g}(\mathbf{q})$ is the force vector associated with the potential energy, and $\boldsymbol{\tau}$ are the motor torque inputs. The kinematic and dynamic parameters of the KUKA LBR iiwa in (2) are taken from [40]. Since the mass matrix $\mathbf{M}(\mathbf{q})$ is invertible, (2) is rewritten in the state-space form

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q})) \end{bmatrix}, \tag{3}$$

with the system state $\mathbf{x}^{\mathrm{T}} = [\mathbf{q}^{\mathrm{T}}, \dot{\mathbf{q}}^{\mathrm{T}}]$. The kinematic and dynamic limits of the robot [22] are summarized in Tab. 2. All limits are symmetric w.r.t. zero, i.e. $\underline{q_i} = -\overline{q_i}$, $\underline{\dot{q}_i} = -\overline{\dot{q}_i}$, and $\underline{\tau_i} = -\overline{\tau_i}$.

To reduce the complexity of the system dynamics (2), the vector of joint acceleration $\ddot{\mathbf{q}}$ is utilized as a new control input for planning a trajectory in Section 2.3, i.e., $\mathbf{u} = \ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}))$. Hence, the system dynamics (3) is rewritten in the compact form

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{u} \end{bmatrix}. \tag{4}$$

2.2. Analytical inverse kinematics

Typically in manipulation tasks, the desired end-effector pose for a point-to-point motion is given in the 6D Cartesian space in the form, cf. (1)

$$\mathbf{T}_{0,d}^e = \begin{bmatrix} \mathbf{R}_{0,d}^e & \mathbf{p}_{t,d} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5}$$

To compute the robot joint configuration $\mathbf{q}$ from a desired end-effector pose $\mathbf{T}^e_{0,d}$, the inverse kinematics (IK) of the robot has to be solved. In the following, an inverse kinematics solution with redundancy parmeters tailored to the non-offset 7-DoF robot KUKA LBR iiwa 14 R820 is shortly revisited, see, e.g., [37]. Similar to [21, 9], the redundancy parameters of this robot are chosen as the binary vector $\mathbf{j}_c^{\mathrm{T}} = [j_s, j_e, j_w]$ and the angle φ, which are introduced below.

With a given end-effector pose $\mathbf{T}^e_{0,d}$, the position of the robot wrist $\mathbf{p}_w$ in the world frame is fixed and is computed as

$$\mathbf{p}_w = \mathbf{p}_{t,d} - \mathbf{R}^7_{0,d} \begin{bmatrix} 0 & 0 & d_7 + d_t \end{bmatrix}^{\mathrm{T}}, \tag{6}$$

with the distance from the wrist point to the end effector of the robot $d_7 + d_t$. The vector $\mathbf{p}_{sw}$ from the fixed shoulder position $\mathbf{p}_s = \begin{bmatrix} 0 & 0 & d_1 + d_2 \end{bmatrix}^{\mathrm{T}}$ to the wrist position $\mathbf{p}_w$ is expressed as $\mathbf{p}_{sw} = \mathbf{p}_w - \mathbf{p}_s$. Using the law of cosines in the triangle formed by the shoulder, elbow, and wrist, the joint position q_4 is immediately calculated as

$$q_4 = j_e \arccos\left(\frac{|\mathbf{p}_{sw}|^2 - d_{se}^2 - d_{ew}^2}{2 d_{se} d_{ew}}\right), \tag{7}$$

where $d_{se} = d_3 + d_4$ is the distance from the shoulder to the elbow and $d_{ew} = d_5 + d_6$ is the distance from the elbow to the wrist. In (7), the binary redundancy parameter $\mathbf{j}_e \in \{-1, 1\}$ distinguishes between the elbow-up and the elbow-down configuration.

The constellation of the shoulder, elbow, and wrist position forms two triangles of which the sides have a constant length for a given end-effector pose. Further, these three points and the two triangles lie on a plane, denoted as arm plane, which can be rotated around the vector $\mathbf{p}_{sw}$ resulting in two cones, see Fig. 3. Thereby, the elbow position p_e always stays on the perimeter of the cone bases. As a result, the robot can perform self-motions by moving the elbow on this perimeter. To this end, an arm angle φ is introduced as a redundancy parameter, referring to the angle between a reference arm with the special configuration $q_{3,n} = 0$ (red lines in Fig. 3), and the actual arm plane (blue lines in Fig. 3). Here and in the following, the index n refers to the reference arm configuration. The actual elbow orientation $\mathbf{R}^4_0$ is equivalent to rotating the orientation of the reference elbow orientation $\mathbf{R}^4_{0,n}$ about the shoulder-wrist vector $\mathbf{p}_{sw}$ by φ, i.e.

$$\mathbf{R}^4_0 = \mathbf{R}_\varphi \mathbf{R}^4_{0,n}, \tag{8}$$

with Rodrigues' formula [26]

$$\mathbf{R}_\varphi = \mathbf{I}_{3\times3} + [\mathbf{p}_{sw}]_\times \sin\varphi + [\mathbf{p}_{sw}]_\times^2 (1 - \cos\varphi) \ , \tag{9}$$

where $\mathbf{I}_{3\times3}$ is the identity matrix, and $[\mathbf{a}]_\times$ denotes the skew-symmetric matrix of the vector $\mathbf{a}$. Since q_4 remains unchanged between the reference arm configuration and the actual arm configuration for a given end-effector pose, (8) leads to

$$\mathbf{R}^4_3 = \mathbf{R}^4_{3,n} \tag{10a}$$

$$\mathbf{R}^3_0 = \mathbf{R}_\varphi \mathbf{R}^3_{0,n} \tag{10b}$$

$$\begin{aligned} \mathbf{R}^7_4 &= (\mathbf{R}^3_0 \mathbf{R}^4_3)^{\mathrm{T}} \mathbf{R}^7_{0,d} \\ &= (\mathbf{R}_\varphi \mathbf{R}^3_{0,n} \mathbf{R}^4_{3,n})^{\mathrm{T}} \mathbf{R}^7_{0,d} \end{aligned} \tag{10c}$$

Note that $\mathbf{R}^3_{0,n}$ depends only on the joint angles $q_{1,n}$ and $q_{2,n}$, since $q_{3,n} = 0$ in the reference configuration. The joint angles $q_{1,n}$ and $q_{2,n}$, shown in Fig. 4, are simply found as

$$q_{1,n} = \arctan2(p_{sw,x}, p_{sw,y}) \tag{11a}$$

$$q_{2,n} = \arctan2\Big(\sqrt{(p_{sw,x})^2 + (p_{sw,y})^2}, p_{sw,z}\Big) + \gamma \ , \tag{11b}$$

with $\mathbf{p}_{sw}^{\mathrm{T}} = [p_{sw,x}, p_{sw,y}, p_{sw,z}]$, and

$$\gamma = j_e \arccos\left(\frac{d_{se}^2 + |\mathbf{p}_{sw}|^2 - d_{ew}^2}{2 d_{se} |\mathbf{p}_{sw}|}\right).$$

Note that $\mathbf{R}^3_0$ and $\mathbf{R}^7_4$ can be directly computed using (10b), (10c) and Tab. 1. Analytically, the rotation matrices $\mathbf{R}^3_0$ and $\mathbf{R}^7_4$ result from Tab. 1 in the form

$$\mathbf{R}^3_0 = \begin{bmatrix} * & * & \cos q_1 \sin q_2 \\ * & * & \sin q_1 \sin q_2 \\ -\sin q_2 \cos q_3 & \sin q_2 \sin q_3 & \cos q_2 \end{bmatrix} \tag{12a}$$

$$\mathbf{R}^7_4 = \begin{bmatrix} * & * & \cos q_5 \sin q_6 \\ -\sin q_6 \cos q_7 & \sin q_6 \sin q_7 & \cos q_6 \\ * & * & -\sin q_5 \sin q_6 \end{bmatrix}, \tag{12b}$$

where the elements written as $*$ are omitted for brevity. From (12), the joint angles of the redundant manipulator are computed in a straightforward manner

$$\begin{aligned} q_1 &= \arctan2(\mathbf{R}^3_0[2,3], \mathbf{R}^3_0[1,3]) \\ q_2 &= j_s \arccos(\mathbf{R}^3_0[3,3]) \\ q_3 &= \arctan2(\mathbf{R}^3_0[3,2], -\mathbf{R}^3_0[3,1]) \\ q_5 &= \arctan2(-\mathbf{R}^7_4[3,3], \mathbf{R}^7_4[1,3]) \\ q_6 &= j_w \arccos(\mathbf{R}^7_4[2,3]) \\ q_7 &= \arctan 2(\mathbf{R}^7_4[2,2], -\mathbf{R}^7_4[2,1]) \ , \end{aligned} \tag{13}$$

where $j_s, j_w \in \{-1, 1\}$ are the remaining binary redundancy parameters and $\mathbf{R}[i, j]$ is the matrix element of the i-th row and j-th column of $\mathbf{R}$.

In summary, the parameterization of the inverse kinematics solution uses the three binary variables $\mathbf{j}_c^{\mathrm{T}} = [j_s, j_e, j_w]$ and the arm angle φ in (7) and (13) as redundancy parameters to determine a unique joint configuration $\mathbf{q}$ for a desired end-effector pose $\mathbf{T}^e_{0,d}$. In Fig. 3, the blue lines illustrate a possible robot configuration with $\mathbf{j}_c = [1, -1, 1]^{\mathrm{T}}$ that is rotated by $\varphi = 95°$ from the reference arm plane, drawn with red lines. To this end, by combining (7) and (13), the unique analytical inverse kinematics of the KUKA LBR iiwa 14 reduces to the compact form

$$\mathbf{q} = \mathrm{AIK}(\mathbf{T}^e_{0,d}, \mathbf{j}_c, \varphi), \tag{14}$$

with the redundancy parameters $\mathbf{j}_c \in \{1, -1\}^3$ and $\varphi \in [0, 2\pi]$.

2.3. Point-to-point trajectory optimization

In the point-to-point (PTP) trajectory planning, a desired trajectory $\boldsymbol{\xi}^*(t) = [\mathbf{x}^*(t), \mathbf{u}^*(t)]^{\mathrm{T}}$, $t \in [t_0, t_F]$ for the robotic system (4) is planned from an initial configuration $\boldsymbol{\xi}^*(t_0) =$

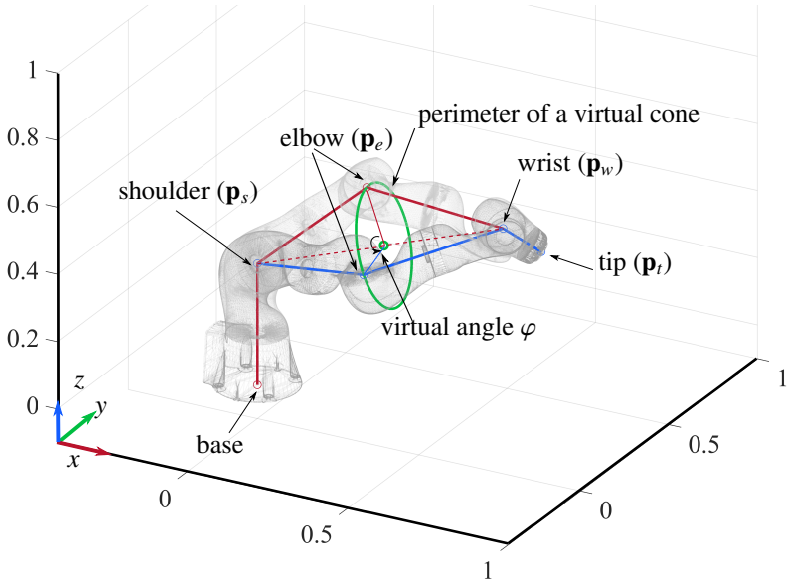

Figure 3: Two configurations of the KUKA iiwa at the same pose are illustrated in red and blue lines. The green rim indicates the virtual movement of the elbow position w.r.t the specific end-effector pose. The red lines and blue lines illustrate the robot at the arm angle $\varphi = 0$ and $\varphi = 95°$, respectively.

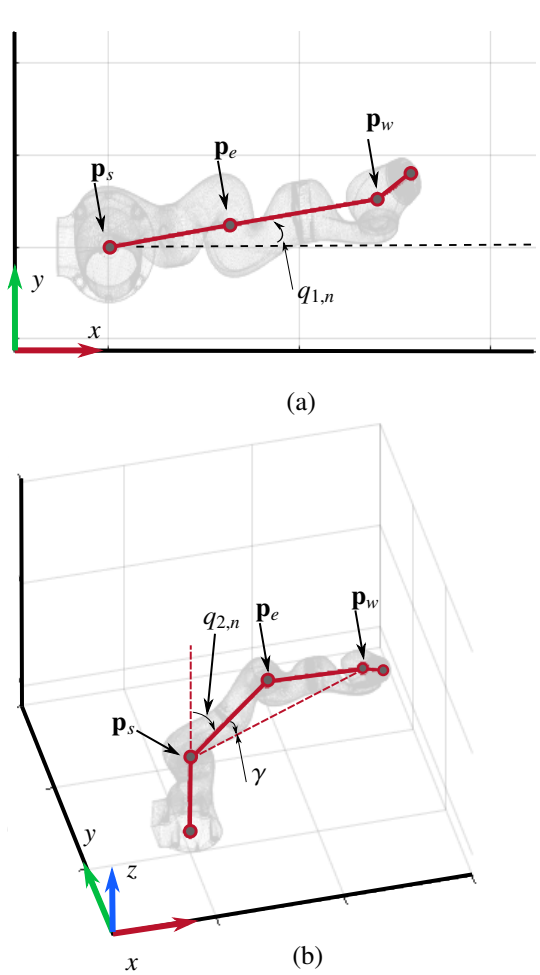

Figure 4: The redundant manipulator ($q_{3,n} = 0$) in the xy-plane (a) and in the 3D xyz-plane (b). The shoulder, elbow, and wrist positions are colinear in (a).

$[\mathbf{x}_{t_0}, \mathbf{u}_{t_0}]^\mathrm{T}$ to a target configuration $\boldsymbol{\xi}^*(t_F) = [\mathbf{x}_{t_F}, \mathbf{u}_{t_F}]^\mathrm{T}$. The target configuration has to satisfy the forward kinematics relation for the desired end-effector pose $\mathbf{T}^e_{0,d}$, see (1)

$$\mathbf{T}^e_{0,d} - \mathrm{FK}(\mathbf{q}_{t_F}) = \mathbf{0} \tag{15}$$

Without loss of generality, the initial time t_0 is chosen as $t_0 = 0$. Furthermore, the target configuration is assumed to be a stationary point $\mathbf{x}_{t_F}^\mathrm{T} = [\mathbf{q}_{t_F}^\mathrm{T}, \mathbf{0}^\mathrm{T}]$.

The PTP trajectory planning is formulated as optimization problem using the direct collocation method, see, e.g., [2], by discretizing the trajectory $\boldsymbol{\xi}(t), t \in [0, t_F]$, with $N+1$ grid points and solving the resulting static optimization problem

$$\min_{\boldsymbol{\xi}} J(\boldsymbol{\xi}) = t_F + \frac{1}{2} h \sum_{k=0}^{N} \mathbf{u}_k^\mathrm{T} \mathbf{R} \mathbf{u}_k \tag{16a}$$

$$\text{s.t. } \mathbf{x}_{k+1} - \mathbf{x}_k = \frac{1}{2} h \begin{bmatrix} \dot{\mathbf{q}}_{k+1} + \dot{\mathbf{q}}_k \\ \mathbf{u}_{k+1} + \mathbf{u}_k \end{bmatrix} \tag{16b}$$

$$\mathbf{x}_0 = \mathbf{x}_{t_0}, \quad \mathbf{x}_N = \mathbf{x}_{t_F} \tag{16c}$$

$$\underline{\mathbf{x}} \le \mathbf{x}_k \le \overline{\mathbf{x}} \tag{16d}$$

$$\underline{\boldsymbol{\tau}} \le \mathbf{M}(\mathbf{q}_k)\mathbf{u}_k + \mathbf{C}(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + \mathbf{g}(\mathbf{q}_k) \le \overline{\boldsymbol{\tau}} \tag{16e}$$

$$k = 0, \ldots, N$$

for the optimal trajectory

$$(\boldsymbol{\xi}^*)^\mathrm{T} = [t_F^*, (\mathbf{x}_0^*)^\mathrm{T}, \ldots, (\mathbf{x}_N^*)^\mathrm{T}, (\mathbf{u}_0^*)^\mathrm{T}, \ldots, (\mathbf{u}_N^*)^\mathrm{T}], \tag{17}$$

with the time step $h = t_F/N$. Note that the final time t_F^* in (17) denotes the optimal duration of the trajectory from the initial state $\mathbf{x}_{t_0}$ to the target state $\mathbf{x}_{t_F}$. In addition, $\mathbf{R}$ is a positive definite weighting matrix for the input $\mathbf{u}$ which also weighs the tradeoff between the cost of the duration and the smoothness of the trajectory. The system dynamics (4) is approximated by the trapezoidal rule in (16b). Moreover, $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ in (16d) denote the

symmetric lower and upper bounds of the state, respectively, and (16c) considers the upper and lower torque limit $\overline{\boldsymbol{\tau}}$ and $\underline{\boldsymbol{\tau}}$.

It should be noted that (16e) is a computationally expensive inequality constraint, mainly because of the large expressions in the Coriolis matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$. Indeed, the Coriolis matrix is often neglected in industrial applications [3]. To still consider the influence of the Coriolis matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ for the torque limits, the range of values of the term $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ is investigated for the KUKA LBR iiwa 14 R820 using a Monte Carlo simulation. In this simulation, 10^8 uniformly distributed random state vectors $\mathbf{x}$ are selected from the admissible operating range, see Tab. 2. This simulation shows that the values of $\mathbf{C}\dot{\mathbf{q}}$ are between $\overline{\mathbf{c}}^{\mathrm{T}} = [6, 8, 3, 4, 1, 1, 0.1]$ N m and $\underline{\mathbf{c}}^{\mathrm{T}} = -[6, 7, 3, 4, 1, 1, 0.1]$ N m, which is much smaller than the torque limits of the motor. Although the influence of the Coriolis matrix on the dynamics of the overall system is not significant, it is still advantageous to consider these physical limits in the optimization problem (16). To this end, the costly inequality condition (16e) is replaced by

$$\underline{\boldsymbol{\tau}} - \underline{\mathbf{c}} \leq (\mathbf{M}(\mathbf{q}_k)\mathbf{u}_k + \mathbf{g}(\mathbf{q}_k)) \leq \overline{\boldsymbol{\tau}} - \overline{\mathbf{c}} \,. \tag{18}$$

The optimal trajectory is computed by solving the static optimization problem (16a)-(16d) and (18) using Interior Point OPTimize (IPOPT), an open-source package based on the interior point method (IPM) for large-scale nonlinear programming, see, e.g., [43].

2.4. Optimal target configuration $\mathbf{q}_{t_F}$

In this section, the optimal choice for the target configuration $\mathbf{q}_{t_F}$ is discussed. The inverse kinematics for a redundant robot does not yield a unique joint configuration $\mathbf{q}_{t_F}$, as presented in Section 2.2. Moreover, choosing an unsuitable target configuration $\mathbf{q}_{t_F}$ may cause the trajectory optimization (16) to fail or deliver poor results.

For redundant robots, there is an infinite number of joint configuration solutions $\mathbf{q}_{t_F}$ for a desired end-effector pose $\mathbf{T}^e_{0,d}$. Therefore, two criteria for selecting the best inverse kinematics solution, i.e. the manipulability and closeness, are introduced in the following and an optimization problem is formulated.

First, the manipulability $m(\mathbf{q})$ [47] is the most popular index used to measure the dexterity of a robot for a specific joint configuration $\mathbf{q}$. It is defined as

$$m(\mathbf{q}) = \sqrt{\det\left(\mathbf{J}(\mathbf{q})\mathbf{J}^{\mathrm{T}}(\mathbf{q})\right)} \,, \tag{19}$$

where the geometric manipulator Jacobian $\mathbf{J}(\mathbf{q})$ takes the form

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_v(\mathbf{q}) \\ \mathbf{J}_\omega(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \mathbf{p}_t(\mathbf{q})}{\partial \mathbf{q}} \\ \dfrac{\partial \boldsymbol{\omega}_t(\mathbf{q})}{\partial \mathbf{q}} \end{bmatrix}. \tag{20}$$

In (20), $\boldsymbol{\omega}_t$ is the angular velocity of the end effector described in the frame $\mathcal{O}_0$, which is computed by

$$[\boldsymbol{\omega}_t]_\times = \dot{\mathbf{R}}^7_0(\mathbf{q})\left(\mathbf{R}^7_0(\mathbf{q})\right)^{\mathrm{T}}. \tag{21}$$

To reduce the computational burden of (19) due to the computation of the determinant, an analytical expression of the manipulability is derived, which is given in the appendix.

Second, to consider the closeness between the inverse kinematics solution $\mathbf{q}$ and the initial joint configuration $\mathbf{q}_0$ of the robot, the L_∞-norm $\|.\|_\infty$ is employed to find the largest deviation between these two joint space configurations. Here, the closeness is given by

$$c(\mathbf{q}) = \|\mathbf{q}_0 - \mathbf{q}\|_\infty, \tag{22}$$

where $\mathbf{q}_{t_0}$ is the initial joint configuration of the initial state $\mathbf{x}_{t_0} = [\mathbf{q}^{\mathrm{T}}_{t_0}, \mathbf{0}]$.

Next, the two criteria (19) and (22) are considered in an optimization problem to choose the best target configuration $\mathbf{q}_{t_F}$ for a given target pose $\mathbf{T}^e_{0,d}$. To solve this problem, according to (14), the redundancy parameters of the inverse kinematics $\mathbf{j}_c$ and φ have to be determined. Since there are three binary redundancy parameters in $\mathbf{j}_c$, $2^3 = 8$ different values are contained in the set $\mathcal{X}_{\mathbf{j}_c} = \{\mathbf{j}_{c,i} \mid i = 1, ..., 8\}$. Additionally, the arm angle $\varphi \in [0, 2\pi]$ is equidistantly discretized with the grid points

$$\mathcal{X}_\varphi = \left\{ j\frac{2\pi}{n_\varphi} \;\middle|\; j = 1, ..., n_\varphi \right\} .$$

The following optimization problem is solved to find the best target configuration $\mathbf{q}_{t_F} = \mathbf{q}^*_{i,j}$ as well as its corresponding redundancy parameters $\mathbf{j}^*_c$ and the virtual angle φ^* for the desired pose $\mathbf{T}^e_{0,d}$

$$\underset{\mathbf{q}^*_{i,j}\mathbf{j}^*_c,\varphi^*}{\arg\min} \left\{ J_{IK}(\mathbf{q}_0, \mathbf{q}_{i,j}) \right\} \begin{array}{l} i \in \{1, ..., 8\} \\ j \in \{1, ..., n_\varphi\} \end{array} \tag{23a}$$

$$\text{s.t. } J_{IK}(\mathbf{q}_0, \mathbf{q}_{i,j}) = \frac{\omega_m}{m(\mathbf{q}_{i,j})} + \omega_c c(\mathbf{q}_0, \mathbf{q}_{i,j}) \tag{23b}$$

$$\mathbf{q}_{i,j} = \mathrm{AIK}(\mathbf{T}^e_{0,d}, \mathbf{j}_{c,i}, \varphi_j) \tag{23c}$$

$$\underline{\mathbf{q}} \leq \mathbf{q}_{i,j} \leq \overline{\mathbf{q}} \;, \tag{23d}$$

with the user-defined weighting parameters $\omega_m > 0$ and $\omega_c > 0$.

3. Framework for learning redundancy parameters

The cost function (23b) and the inverse kinematics (23c) are nonlinear and discontinuous functions with many local minima, which is illustrated on the right-hand side of Fig. 5 for an example joint configuration $\mathbf{q}$. Therefore, to find the global optimum, the optimization problem (23) has to be solved by exhaustive search, which is a time-consuming process since (23c) has to be evaluated $8n_\varphi$ times. To significantly reduce the computational effort for this step, a neural network (NN) is presented in this section to quickly determine the joint configuration $\mathbf{j}^*_c$ and narrow down the search space for the arm angle φ^* for a desired end-effector pose $\mathbf{T}^e_{0,d}$ and the given initial configuration $\mathbf{q}_0$.

First, the generation of the database to train the NN for learning the redundancy parameters $\mathbf{j}_c$ and φ is introduced. Then, the network architecture of this NN is presented in the next step.

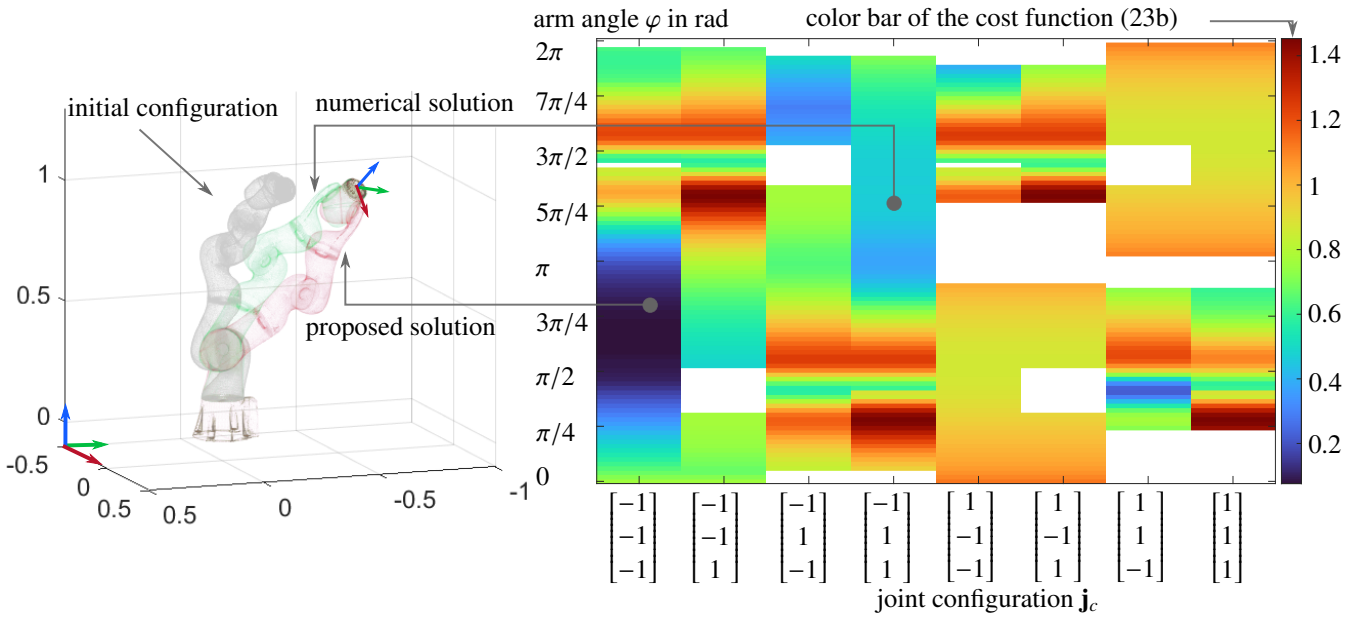

Figure 5: The color map of the cost function (23b) for the initial configuration of the robot $\mathbf{q}_0$ (in gray color) and the position of the end effector (the RGB triad) is shown on the right-hand side of this figure. The white color regions depict infeasible joint configurations. The robot target configuration $\mathbf{q}_{t_F}$ computed by the proposed NN is shown in red color on the left-hand side. This achieves a very small value of the cost function (23b) (≈ 0.0717). The target robot configuration calculated using the numerical method [5] is depicted in green. The cost of this configuration is approximately 0.54.

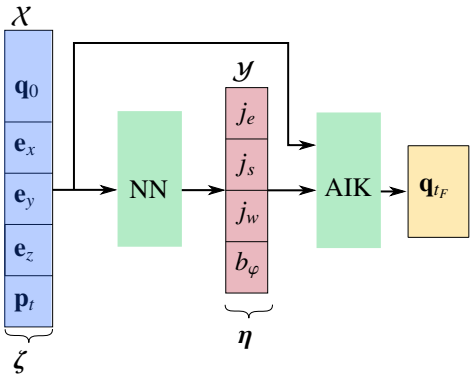

Figure 6: Overview of the proposed framework for learning the redundancy parameters.

3.1. Database generation

For the database generation, N_p pairs of robot initial joint configurations $\mathbf{q}_{0,k}$ and corresponding feasible desired poses $\mathbf{T}_{0,d}^{e,k}$ are randomly selected from a uniform random distribution in the admissible ranges and are stored in the set $\mathcal{X} = \{\boldsymbol{\zeta}_k\} = \{\mathbf{q}_{0,k}, \mathbf{T}_{0,d}^{e,k} \mid k = 1, ..., N_p\}$. For each pair of $\mathbf{q}_{0,k}$ and $\mathbf{T}_{0,d}^{e,k}$, the optimization problem (23) is solved by an exhaustive search to find the global optimum redundancy parameters $\mathbf{j}_c^*$ and φ^* as well as the target configuration $\mathbf{q}_{t_F}^*$. The redundancy parameters are stored in the set $\mathcal{Y} = \{\boldsymbol{\eta}_k\} = \{\mathbf{j}_{c,k}, \varphi_k \mid k = 1, ..., N_p\}$. The database $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ comprises both sets $\mathcal{X}$ and $\mathcal{Y}$. Elements from the set $\mathcal{X}$ are the input to the NN and elements from the set $\mathcal{Y}$ are the corresponding labeled outputs, see Fig. 6.

The input data in the set $\mathcal{X}$ contain redundant data due to the constant bottom row in the desired pose $\mathbf{T}_{0,d}^{e,k}$, see (5). Therefore, only the three basis vectors $\mathbf{e}_{x,k}$, $\mathbf{e}_{y,k}$, and $\mathbf{e}_{z,k}$, of $\mathbf{R}_0^{e,d} = [\mathbf{e}_{x,k}, \mathbf{e}_{y,k}, \mathbf{e}_{z,k}]$ and the position of the end effector $\mathbf{p}_{t,k}$ are considered in the set $\mathcal{X}$. Thus, the input set $\mathcal{X}$ is re-arranged in the form

$$\mathcal{X} = \{\boldsymbol{\zeta}_k \mid k = 1, ..., N_p\},$$

with

$$\boldsymbol{\zeta}_k^{\mathrm{T}} = [(\mathbf{q}_{0,k})^{\mathrm{T}}, (\mathbf{e}_{x,k})^{\mathrm{T}}, (\mathbf{e}_{y,k})^{\mathrm{T}}, (\mathbf{e}_{z,k})^{\mathrm{T}}, (\mathbf{p}_{t,k})^{\mathrm{T}}] \in \mathbb{R}^{19}$$

Since (23b)-(23c) are discontinuous nonlinear functions, see Fig. 5, a complex NN is required to approximate these functions. However, the training and prediction time of such a neural network is very long, making it impossible to be implemented in real time. Thus, the continuous output variable for the predicted arm angle φ is discretized into n_b bins and the bin index b_φ is predicted instead. This helps to reduce the complexity of the problem. This way, the i-th bin contains the range $\left[(i-1)\frac{2\pi}{n_b}, i\frac{2\pi}{n_b}\right], i = 1, ..., n_b$. Consequently, φ_k is replaced by its bin index $b_{\varphi,k} \in \{1, ..., n_b\}$ in the set $\mathcal{Y}$ resulting in $\mathcal{Y} = \boldsymbol{\eta}_k = \{\mathbf{j}_{c,k}, b_{\varphi,k}\}$.

3.2. Network Architecture

The architecture of the proposed NN is shown in Fig. 7. This NN is designed for the two sub-problems, i.e., to learn the joint configuration $\mathbf{j}_c$ and the bin index b_φ of the arm angle φ. Note that the input of the proposed NN is $\boldsymbol{\zeta} \in \mathcal{X}$ and the output is the predicted value $\boldsymbol{\eta}^{\mathrm{T}} = [\mathbf{j}_c^{\mathrm{T}}, b_\varphi] \in \mathcal{Y}$.

First, two fully connected layers of size 32 with a ReLU activation function [23] are utilized, as shown in Fig. 7. Since there are 8 possibilities for choosing $\mathbf{j}_c$, a fully connected layer of size 8 with a softmax activation function [4] is employed to output $\mathbf{j}_c$. The cross-entropy function is used to compute the

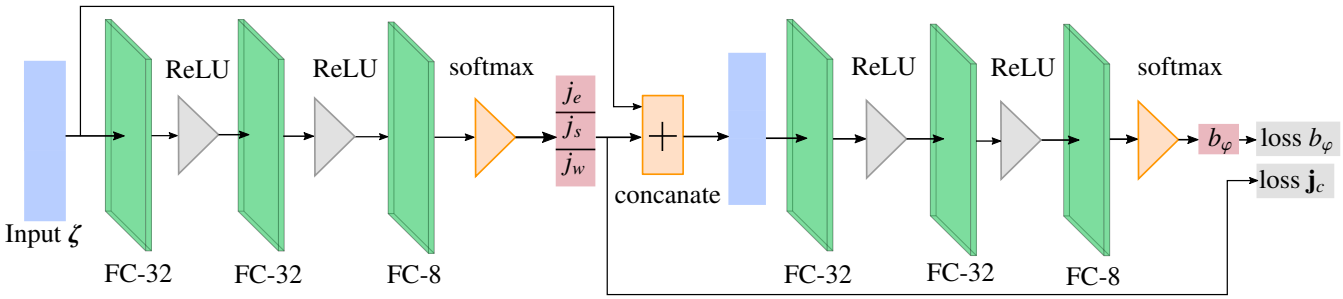

Figure 7: Architecture of the proposed NN for learning the joint configuration $\mathbf{j}_c$ and the bin index b_φ of the virtual angle φ.

loss between the prediction $\hat{\mathbf{j}}_{c,k}$ of the NN and the target value $\mathbf{j}_{c,k}$ in the form

$$\mathrm{L}_{\mathbf{j}_c} = \sum_{k=1}^{M} -\mathbf{j}_{c,k}^{\mathrm{T}} \log(\hat{\mathbf{j}}_{c,k}) + (\mathbf{1} - \mathbf{j}_{c,k})^{\mathrm{T}} \log(\mathbf{1} - \hat{\mathbf{j}}_{c,k}) \ , \qquad (24)$$

where M is the size of the training dataset.

Second, the predicted $\hat{\mathbf{j}}_c$ is concatenated with the input $\boldsymbol{\zeta}$ again as a new input for the second subproblem. Similar to the first subproblem, two fully connected layers of size 32 with a ReLU activation function are used. Subsequently, the fully connected layer of size 8 and the softmax activation function are implemented to predict the bin index b_φ of the arm angle φ. Again, the cross entropy function is used to compute the loss between the predicted value of the bin index $\hat{b_\varphi}$ and the target value b_φ

$$\mathrm{L}_{b_\varphi} = \sum_{k=1}^{M} -b_{\varphi,k} \log(\hat{b}_{\varphi,k}) + (1 - b_{\varphi,k}) \log(1 - \hat{b}_{\varphi,k}). \qquad (25)$$

The employed optimizer of the proposed NN is Adam [19] with a learning rate of 10^{-3}. Furthermore, the L_2 regularization [19] with $\lambda = 10^{-6}$ is added to both loss functions $\mathrm{L}_{\mathbf{j}_c}$ and L_{b_φ} to avoid overfitting [27].

4. Results

The simulation results presented in this section are obtained on a computer with a 3.4 GHz Intel Core i7-10700K and 32 GB RAM. The generated database with $N_p = 10^8$ pairs described in Section 3.1 is randomly shuffled and divided into 3 subsets, i.e., training dataset, validation dataset, and test dataset, which are partitioned as 80%, 10%, and 10% of the generated database, respectively. To speed up the computing time of database generation, C++ code is generated for (23) using MATLAB coder in MATLAB R2021b. Additionally, the analytical expression of the manipulability in the appendix (29) is utilized. The remaining parameters are chosen as $n_\varphi = 100$ and $n_b = 8$. For the database generation, the average computing time of (23) for a given pose and initial joint configuration is approximately 1.5 ms. Since n_φ in (23) is set to 100, the average computing time of the analytical inverse kinematics expression in (23c) is approximately 1.8 µs.

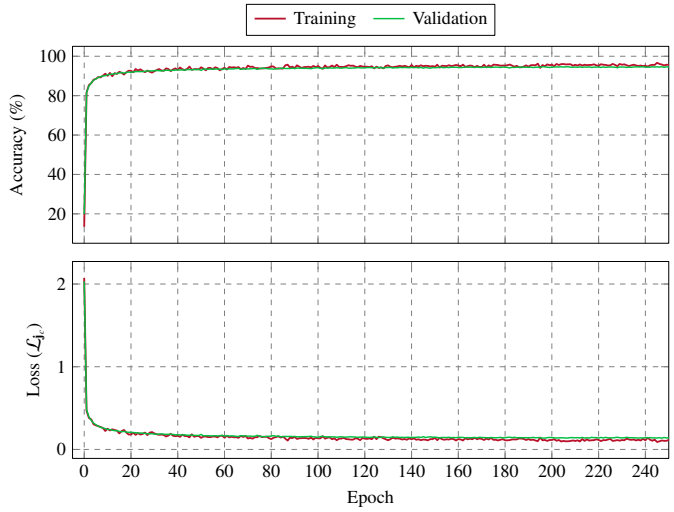

Figure 8: Training and validation accuracy of the joint configuration $\mathbf{j}_c$.

4.1. Statistical information on training the proposed NN

The proposed NN is trained using the open-source software package Keras [13]. To reduce the training time, the CUDA cores of an Nvidia GeForce RTX 3070 are employed. During training, the mini-batch size is set to 2000 and the training data is reshuffled in each epoch.

Fig. 8 shows that the learning accuracy for the joint configuration $\mathbf{j}_c$ of the training dataset and the validation dataset after 500 epochs reaches 96.62% and 95.76%, respectively. Also, the corresponding values of the loss function $\mathrm{L}_{\mathbf{j}_c}$ decreased to 0.1034 and 0.1264, respectively. To further validate the training result, the accuracy of the test dataset with the trained parameters of the proposed NN is approximately 96.49%.

Fig. 9 shows the accuracy of the training dataset and the validation dataset with respect to the bin index b_φ of the arm angle φ. Note that the resulting accuracy is approximately 85.57% for the training dataset and 85.12% for the validation dataset. The values of the loss function L_{b_φ} are approximately 0.32 and 0.38 for the two datasets. To verify the trained parameters of the proposed NN, a consistent accuracy of 84.78% is reported for the test dataset.

For further validation, the proposed NN is compared to the performance of four well-known algorithms, i.e., naive Bayes

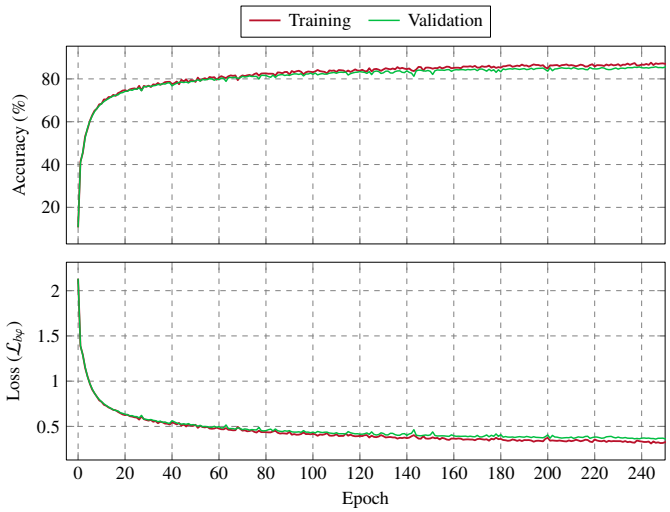

Figure 9: Training and validation accuracy of the bin index b_φ.

Table 3: Performance of the prediction with different algorithms

Classifier	Acc. $\mathbf{j}_c$ %	Acc. b_φ %	Time µs	Memory MB
Naive Bayes [15]	57.6	38.2	1.1	76.8
Discriminant Analysis [42]	65.1	40.6	1.23	70.5
Binary Decision Tree [25]	77.8	65.5	0.35	89.6
k-Nearest Neighbor [17]	49.5	40.1	1810	76.8
Proposed NN	96.5	84.8	7.35	0.17

classifier [15], discriminant analysis classifier [42], binary decision tree classifier [25] and k-nearest neighbor classifier [17]. Similar to the proposed NN, each classifier takes the input $\boldsymbol{\zeta} \in \mathcal{X}$ and outputs the prediction $\boldsymbol{\eta} \in \mathcal{Y}$. The statistical performance of the four algorithms and the proposed NN is shown in Tab. 3. Among the above algorithms, the binary decision tree classifier achieves the highest prediction accuracy for $\mathbf{j}_c$ and b_φ, i.e., 77.8% and 65.5%, respectively. Moreover, the average execution time of this classifier is approximately 0.35 µs, i.e., the fastest algorithm. However, the prediction accuracy of the proposed NN is still significantly higher compared to the binary decision tree classifier. Another aspect is the memory consumption, which is with 0.17 MB much less compared to over 70 MB for each of the four other algorithms. This is reasonable since in the proposed NN, the memory consumption is mainly used for storing the network parameters.

The average NN execution time for the prediction of $\mathbf{j}_c$ and b_φ is about 7.35 µs. Also, the average computing of the analytical inverse kinematics with the given $\mathbf{j}_c$ and b_φ is about 2 µs. Thus, the proposed NN provides the possibility to compute a good IK solution with respect to the two criteria, i.e., manipulability (19) and closeness (22), in real time.

4.2. *Performance of the proposed NN in the framework of trajectory optimization*

To verify the efficiency of the proposed NN, the example task of planning a PTP trajectory from the initial configuration

$$\begin{aligned} \mathbf{q}_0^{\mathrm{T}} &= [-1.5, -0.1, 0.3, 0.7, 0.5, -0.6, 1.4]\ \text{rad}, \\ \mathbf{j}_{c,0}^{\mathrm{T}} &= [-1, 1, -1], \\ \varphi_0 &= 3.21\ \text{rad} \end{aligned}$$

to the target pose

$$\mathbf{T}_{0,d}^{e} = \begin{bmatrix} 0.863 & 0.262 & -0.433 & -0.55 \\ 0.003 & 0.853 & 0.522 & 0.160 \\ 0.505 & -0.451 & 0.735 & 1.049 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

is considered.

First, the comparison between the well-known damped least-squares inverse kinematics solution [26, 5] and the proposed algorithm is depicted in Fig. 5. On the right-hand side of Fig. 5, a color map of (23b) is depicted where the x-axis comprises the 8 possible joint configurations $\mathbf{j}_c \in \mathcal{X}_{\mathbf{j}_c}$ and the y-axis contains the $n_\varphi = 100$ arm angles $\varphi \in \mathcal{X}_\varphi$. Using the network architecture of Fig. 7 with $n_b = 8$, the proposed NN takes about 7.35 µs to predict the joint configuration $\mathbf{j}_c = [-1, -1, -1]^{\mathrm{T}}$ and the bin $b_\varphi = 3$, i.e. $\varphi \in [\pi/2, 3\pi/4]$. To find the optimum value for the arm angle φ inside the predicted bin, (14) and (23a) are evaluated on an equidistant grid for $\varphi \in [\pi/2, 3\pi/4]$ with n_φ/n_b grid points. This way, the effort to solve the optimization problem (23) reduces from $8n_\varphi = 800$ to $n_\varphi/n_b \approx 13$ evaluations of (14) and (23b). Since the analytical manipulability expression (29) is used in (23b), the computing time of (23b) is approximately 0.15 µs, which is much smaller than (14). Thus, the total execution time for computing the optimal target configuration $\mathbf{q}_{t_F}$ is approximately 32 µs including the computing time of (14) of 2 µs. On the other hand, the damped least-squares method in this example requires 17 iterations to find the solution of the inverse kinematics with a tolerance of 10^{-8}. The computing time of the numerical method is approximately 3 ms.

On the left-hand side of Fig. 5, the computed target configurations for the given desired target pose $\mathbf{T}_{0,d}^{e}$ are

$$\begin{aligned} \mathbf{q}_{t_F,A}^{\mathrm{T}} &= [-0.55, -0.96, -0.71, -0.78, -0.45, -0.8, 1.55]\ \text{rad}, \\ \mathbf{j}_{c,A}^{\mathrm{T}} &= [-1, -1, -1], \\ \varphi_A &= 2.17\ \text{rad} \end{aligned} \tag{27}$$

for the proposed algorithm (red color), and

$$\begin{aligned} \mathbf{q}_{t_F,N}^{\mathrm{T}} &= [-0.7, -0.45, 1.1, 0.78, 0.43, 0.81, -0.82]\ \text{rad}, \\ \mathbf{j}_{c,N}^{\mathrm{T}} &= [-1, 1, 1], \\ \varphi_A &= 3.8\ \text{rad} \end{aligned} \tag{28}$$

for the damped least-squares method (green color). It is obvious that in this example the joint configuration solutions of

the two methods $\mathbf{j}_{c,A}$ and $\mathbf{j}_{c,N}$ are different from the initial joint configuration $\mathbf{j}_{c,0}$. The proposed solution has a slightly higher manipulability measure (29) of 0.061 compared to the manipulability measure of 0.045 of the numerical solution. The closeness value (22) of the proposed solution is 1.49 which is significantly smaller than the closeness value of 2.23 of the numerical solution.

To further demonstrate the effectiveness of the proposed IK approach, the two target configurations obtained by the proposed algorithm and the numerical method are used in the trajectory optimization framework detailed in Section 2. The nonlinear optimization problem (16) is solved using the interior point solver IPOPT [43] together with the linear solver MA27 [8]. To increase the computational speed, the gradient and the numerical Hessian are computed using the BFGS method [24] and provided to the IPOPT solver. The trajectory in (16) is discretized with $N = 50$ collocation points, giving a total of 1051 optimization variables. For this comparison, the same initial configuration $\mathbf{q}_0$ and two different target configurations $\mathbf{q}_{t_F,A}$ according to (27) and $\mathbf{q}_{t_F,N}$ according to (28) of the pose $\mathbf{T}^e_{0,d}$ from (26) are used. While the computing time of the optimization (16) for both target configurations is almost the same (55 ms), the time for moving to the target configuration of the proposed algorithm $\mathbf{q}_{t_F,A}$ is 3.83 s compared to 4.03 s of the numerical solution $\mathbf{q}_{t_F,N}$. Moreover, the cost function in (16a) with $\mathbf{q}_{t_F,A}$ and $\mathbf{q}_{t_F,N}$ is 4.7 and 5.03, respectively.

Finally, a Monte Carlo simulation is performed to validate the efficiency of the proposed NN in the PTP trajectory optimization. To this end, 10^5 pairs of initial robot configurations $\mathbf{q}_0$ and target poses $\mathbf{T}^e_{0,d}$ are randomly selected from a uniform random distribution in the admissible ranges. Then, the proposed NN and the numerical IK are used to determine the target joint configuration and for each target configuration, an optimal trajectory is calculated using (16). The statistical results are summarized in Tab. 4. While the computing times of (16) utilizing the target configuration of the proposed algorithm $\mathbf{q}_{t_F,A}$ and the numerical IK $\mathbf{q}_{t_F,N}$ are nearly the same ($\approx$ 30 ms), the average optimal trajectory time using the proposed algorithm is slightly better, i.e. 4.52 s compared to 5.39 s.

Table 4: Performance of the proposed NN and the numerical IK [26] in the trajectory optimization framework

	Proposed NN	Numerical IK [26]
Avg. t_F (s)	4.52 ± 1.93	5.39 ± 2.6
Cost value of (16a)	5.75 ± 2.79	6.69 ± 3.29
Num. of failed IK	0	13896
Num. of failed PTP	554	1588
Success rate	99.5%	84.5%
Avg. comp. time (ms) of (16)	28.9 ± 13	30.3 ± 19

Since the solution of the numerical IK depends on the initial guess, 13896 test cases fail to converge to feasible target configurations. Note that the maximum number of iterations for the numerical IK is 50. Additionally, after excluding 13896 failed cases, 1588 test cases are not valid to plan the PTP trajectory using (16). Note that these test cases fail because of violating the iteration limit, i.e., 100 iterations, which is set in the IPOPT solver. The overall success rate by using the numerical IK is approximately 84.5%. On the other hand, for the proposed algorithm, in all the test cases, a feasible target configuration is found. Only 554 test cases fail during the planning of the PTP trajectory due to the iteration limit of the IPOPT solver. Overall, the proposed NN outperforms the numerical IK by achieving a success rate of 99.5%.

5. Conclusions

In this work, a machine learning-based approach for the inverse kinematics (IK) of kinematically redundant robots is presented, which is suitable for trajectory planning in highly dynamic real-time applications like human-robot object handovers or robotic object catching. In this approach, the optimal redundancy parameters are predicted by a neural network (NN) according to the application-specific criteria, closeness to the initial robot configuration and manipulability at the target pose. Redundancy parameters, i.e. a virtual arm angle and binary variables describing the joint configurations, resolve the non-uniqueness of the analytical IK of redundant robots and allow for a unique mapping between the target pose and the joint configuration. Since a NN is employed, the proposed framework can be applied to different collaborative robots, e.g., KUKA LBR iiwa 14 R820, Franka Emika Panda, OB7, of which the analytical IK can be parameterized by redundancy parameters. The NN used in the proposed framework outperforms classical classification algorithms in terms of accuracy and the prediction run time. A Monte Carlo simulation of 10^5 random pairs of an initial configuration and a target pose validates the proposed algorithm in the context of point-to-point (PTP) trajectory optimization. The proposed method succeeds in 99.5% of the test cases to find a feasible target configuration while achieving a shorter optimal time of the trajectory from the initial to the target pose on average compared to using a numerical IK method at a significantly shorter computing time ($\approx$ 32 µs for the proposed IK compared to $\approx$ 3 ms for the numerical IK). Thus, the proposed framework is perfectly suited for real-time applications.

In future works, this machine learning-based framework will be applied to dynamic human-robot handover tasks.

Appendix

The square of the manipulability (19) of the KUKA LBR iiwa 14 R820 [22] reads as

$$
\begin{aligned}
m^2(\mathbf{q}) = {} & 2d_{se}^2 d_{ew}^2 \sin^2(q_4)\cdot \\
& \Big[d_{se}^2 \sin^2(q_2)\sin^2(q_4)\cos^2(q_5)\cos^2(q_6) + \\
& d_{ew}^2 \cos^2(q_2)\cos^2(q_3)\sin^2(q_4)\sin^2(q_6) + \\
& (d_{se}^2 + 2d_{se}d_{ew}\cos(q_4) - d_{ew}^2)\sin^2(q_2)\sin^2(q_6) + \\
& \frac{1}{2}(d_{se}^2\cos(q_4) + d_{se}d_{ew})\sin^2(q_2)\sin(q_4)\cos(q_5)\sin(2q_6) + \\
& \frac{1}{2}(d_{ew}^2\cos(q_4) + d_{se}d_{ew})\sin(2q_2)\cos(q_3)\sin(q_4)\sin^2(q_6)\Big],
\end{aligned} \tag{29}
$$

where $d_{se} = d_2 + d_3$ and $d_{ew} = d_4 + d_5$.

Conflict of interest

The authors have no conflicts of interest to declare.

References

[1] Betts, J.T., 1998. Survey of numerical methods for trajectory optimization. Journal of Guidance, Control, and Dynamics 21, 193–207.

[2] Betts, J.T., 2010. Practical methods for optimal control and estimation using nonlinear programming. Siam: Philadelphia.

[3] Binder, E.E., Herzog, J.H., 1986. Distributed computer architecture and fast parallel algorithms in real-time robot control. IEEE Transactions on Systems, Man, and Cybernetics 16, 543–549.

[4] Bishop, C.M., Nasrabadi, N.M., 2006. Pattern recognition and machine learning. Springer: New York.

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

[6] Cox, D., Little, J., OShea, D., 2013. Ideals, Varieties, and Algorithms: an introduction to computational algebraic geometry and commutative algebra. Springer: Germany.

[7] Diankov, R., Kuffner, J., 2008. Openrave: A planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34 79.

[8] Duff, I.S., 2004. MA57:A Code for the Solution of Sparse Symmetric Definite and Indefinite Systems. ACM Transactions on Mathematical Software 30, 118–144.

[9] Faria, C., Ferreira, F., Erlhagen, W., Monteiro, S., Bicho, E., 2018. Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance. Mechanism and Machine Theory 121, 317–334.

[10] Franka Emka GmbH, [Accessed on 07 October 2022]. the new FRANKA PRODUCTION 3: Think it, make it - the robotic automation tool for everyone. URL: https://www.franka.de/production/.

[11] Gattringer, H., Mueller, A., Oberherber, M., Kaserer, D., 2022. Time-optimal robotic manipulation on a predefined path of loosely placed objects: Modeling and experiment. Mechatronics 84, 102753.

[12] Ghafil, H.N., Jármai, K., 2018. Research and application of industrial robot manipulators in vehicle and automotive engineering, a survey, in: Vehicle and Automotive Engineering 2, Springer International Publishing: Cham. pp. 611–623.

[13] Gulli, A., Pal, S., 2017. Deep learning with Keras. Packt Publishing Ltd: Birmingham.

[14] Han, W., Tedrake, R., 2020. Local trajectory stabilization for dexterous manipulation via piecewise affine approximations, in: IEEE International Conference on Robotics and Automation (ICRA), pp. 8884–8891.

[15] Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H., 2009. The elements of statistical learning: data mining, inference, and prediction. volume 2. Springer: Berlin.

[16] He, Y., Liu, S., 2021. Analytical inverse kinematics for franka emika panda–a geometrical solver for 7-dof manipulators with unconventional design, in: IEEE International Conference on Control, Mechatronics and Automation (ICCMA), pp. 194–199.

[17] Jiang, L., Cai, Z., Wang, D., Jiang, S., 2007. Survey of improving k-nearest-neighbor for classification, in: IEEE International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 679–683.

[18] Kang, M., Cho, Y., Yoon, S.E., 2022. RCIK: Real-Time Collision-Free Inverse Kinematics Using a Collision-Cost Prediction Network. IEEE Robotics and Automation Letters 7, 610–617.

[19] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: Poster Presentations of International Conference on Learning Representations ICLR.

[20] Kraemer, M., Muster, F.I., Roesmann, C., Bertram, T., 2021. An optimization-based approach for elasticity-aware trajectory planning of link-elastic manipulators. Mechatronics 75, 102523.

[21] Kuhlemann, I., Schweikard, A., Jauer, P., Ernst, F., 2016. Robust inverse kinematics by configuration control for redundant manipulators with seven dof, in: IEEE International Conference on Control, Automation and Robotics (ICCAR), pp. 49–55.

[22] KUKA Roboter GmbH, [Accessed on 07 October 2022]. LBR iiwa 7 R800 and LBR iiwa 14 R820 specification URL: https://www.kuka.com/en-at/products/robotics-systems/industrial-robots/lbr-iiwa.

[23] Li, Y., Yuan, Y., 2017. Convergence analysis of two-layer neural networks with relu activation. Advances in neural information processing systems 30, 597–607.

[24] Liu, D.C., Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. Mathematical Programming 45, 503–528.

[25] Loh, W.Y., 2002. Regression tress with unbiased variable selection and interaction detection. Statistica sinica , 361–386.

[26] Lynch, K.M., Park, F.C., 2017. Modern robotics. Cambridge University Press: Cambridge.

[27] Murphy, K.P., 2012. Machine learning: a probabilistic perspective. MIT press: Cambridge.

[28] Neura Robotics GmbH, [Accessed on 07 October 2022]. LARA - Lightweight Agile Robotic Assistant URL: https://neura-robotics.com/products/lara.

[29] Paul, R.P., Shimano, B., 1979. Kinematic control equations for simple manipulators, in: IEEE Conference on Decision and Control (CDC) including the 17th Symposium on Adaptive Processes, pp. 1398–1406.

[30] Pellicciari, M., Berselli, G., Leali, F., Vergnano, A., 2013. A method for reducing the energy consumption of pick-and-place industrial robots. Mechatronics 23, 326–334.

[31] Productive Robotics, Inc., [Accessed on 07 October 2022]. MEET THE OB7 COBOT FAMILY URL: https://www.productiverobotics.com/ob7-products.

[32] Qiu, S., Kermani, M.R., 2021. Precision grasp using an arm-hand system as a hybrid parallel-serial system: A novel inverse kinematics solution. IEEE Robotics and Automation Letters 6, 8530–8536.

[33] Raghavan, M., Roth, B., 1993. Inverse Kinematics of the General 6R Manipulator and Related Linkages. Journal of Mechanical Design 115, 502–508.

[34] Safeea, M., Bearee, R., Neto, P., 2021. A modified dls scheme with controlled cyclic solution for inverse kinematics in redundant robots. IEEE Transactions on Industrial Informatics 17, 8014–8023.

[35] Saut, J.P., Gharbi, M., Cortés, J., Sidobre, D., Siméon, T., 2010. Planning pick-and-place tasks with two-hand regrasping, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4528–4533.

[36] Sciavicco, L., Siciliano, B., 1988. A solution algorithm to the inverse kinematic problem for redundant manipulators. IEEE Journal on Robotics and Automation 4, 403–410.

[37] Shimizu, M., Kakuya, H., Yoon, W.K., Kitagaki, K., Kosuge, K., 2008. Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. IEEE Transactions on Robotics 24, 1131–1142.

[38] Siciliano, B., 1990. Kinematic control of redundant robot manipulators:

A tutorial. Journal of intelligent and robotic systems 3, 201–212.
[39] Spong, M., Hutchinson, S., Vidyasagar, M., 2005. Robot Modeling and Control. Wiley: Newyork.
[40] Stürz, Y.R., Affolter, L.M., Smith, R.S., 2017. Parameter identification of the KUKA LBR iiwa robot including constraints on physical feasibility. IFAC PapersOnLine 50, 6863–6868.
[41] Taki, S., Nenchev, D., 2014. A novel singularity-consistent inverse kinematics decomposition for srs type manipulators, in: IEEE International Conference on Robotics and Automation (ICRA), pp. 5070–5075.
[42] Tharwat, A., 2016. Linear vs. quadratic discriminant analysis classifier: a tutorial. International Journal of Applied Pattern Recognition 3, 145–180.
[43] Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106, 25–57.
[44] Wang, H., Wang, H., Huang, J., Zhao, B., Quan, L., 2019. Smooth point-to-point trajectory planning for industrial robots with kinematical constraints based on high-order polynomial curve. Mechanism and Machine Theory 139, 284–293.
[45] Wiedmeyer, W., Altoé, P., Auberle, J., Ledermann, C., Kröger, T., 2020. A real-time-capable closed-form multi-objective redundancy resolution scheme for seven-dof serial manipulators. IEEE Robotics and Automation Letters 6, 431–438.
[46] Xing, H., Torabi, A., Ding, L., Gao, H., Li, W., Tavakoli, M., 2021. Enhancing kinematic accuracy of redundant wheeled mobile manipulators via adaptive motion planning. Mechatronics 79, 102639.
[47] Yoshikawa, T., 1985. Manipulability of robotic mechanisms. The International Journal of Robotics Research 4, 3–9.
[48] Zhou, K., Ebenhofer, G., Eitzinger, C., Zimmermann, U., Walter, C., Saenz, J., Castaño, L.P., Hernández, M.A.F., Oriol, J.N., 2014. Mobile manipulator is coming to aerospace manufacturing industry, in: IEEE international symposium on robotic and sensors environments (ROSE), pp. 94–99.