2024 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
July 15-19, 2024. Boston, MA, USA

Improving Data-based Trajectory Generation by Quadratic Programming for Redundant Mobile Manipulators*

Alice Hierholz[1], Andreas Gienger[1] and Oliver Sawodny[1]

***Abstract*— Challenges in trajectory generation in robotics such as nonlinearities, non-convex constraints, many optimization variables and redundancy lead to new developments in the use of data-based methods. In this paper, a combination of a data- and model-based approach for trajectory generation for a redundant mobile manipulator with 10 degrees of freedom is presented with the goal of reducing the computational cost and improving scalability. A computationally efficient neural network regression model is proposed which predicts the joint trajectories generated from an optimal control problem considering the system equations, redundancy, singularities, nonlinear kinematic and dynamic constraints as well as collisions. The prediction is then improved through a subsequent quadratic programming with low computational cost, ensuring the compliance with the system equations and increasing the tool-center-point target reaching accuracy.**

I. INTRODUCTION

Trajectory generation in robotics is a topic that lies in the focus of many research projects. It is a highly complex topic, since it has to deal with strong nonlinearities, non-convex constraints, a high number of optimization variables and high degrees of freedom (DOF) up to redundant robots, to name only a few challenges. These challenges lead to a generally high computation time, which means that most algorithms can only be deployed offline.

Data-based approaches offer new opportunities to tackle these challenges, especially through the further development of hardware, and therefore gain popularity in the field of robotics. Much research has been conducted in data-based methods for computing the inverse kinematics (IK) of robots, which is then used for trajectory generation. A neural network (NN) approach is presented in [1] for IK computation of a typical 6-DOF robotic arm. The joint space is divided into different subspaces, with a separate NN being trained for each of them. Additionally a classifier is used to determine joint spaces which can be excluded from the search space to speed up computation time. Data-based IK methods for redundant robots are studied extensively in the literature. In [2] a Generative Adversarial Network is used to learn valid robot configurations of high DOF robots considering additional constraints. The computed configurations are then used as initial value for classic IK solvers, whose computation time is speed up. Another approach is to reduce the IK problem of a redundant robot to one of a non-redundant robot. In [3], this is achieved by parameterizing redundant joints using workspace and joint space clustering, with the clustering being performed by a growing neural gas network.

Another way to use data-based methods is to use their prediction to initialize the used trajectory optimization algorithm. This avoids local minima and reduces the computation time as shown in [4]. The motion planning can also directly be executed by data-based approaches. In [5] an encoder network is used to encode the environment followed by a bidirectional planning network, which iteratively computes the robots path in task space. The algorithm is evaluated for 2D scenarios of mobile robots and 3D scenarios of manipulators and shows very low computation times. A further elaboration of this approach is found in [6]. The encoder network is adapted in order to be more parameter-efficient. This leads to further reduction of the computation time which enables the implementation of the method on resource-limited devices. A recurrent neural network with long short term memory units is used in [7] for iterative bidirectional path planning of robots in static environments with up to 7 DOFs. The usage of convolutional neural networks is shown in [8] for path planning of mobile robots using laser scans of the environment and the desired target position as inputs. A combination of several data-based methods is leveraged in [9]. An autoencoder followed by a dynamics network and a collision network learn a latent space, which is then used by a rapidly-exploring random tree to compute paths in it. Multi-agent optimal control problems (OCPs) are solved in [10] through a NN approximating the OCPs value function only for a subset of the task space. The NN therefore needs to be retrained depending on the desired goal position and the associated subset. The robots investigated include mobile robots for 2D scenarios and quadcopters for 3D scenarios. Learning trajectories of quadcopters with simple linear kinematics is also investigated in [11]. A NN is used to learn a parametric OCP. The predicted trajectory of the NN is then refined by solving a one-step quadratic programming (QP). The basic idea of [11] serves as foundation for the method used in this paper.

The proposed approach in this work consists of the combination of a data-based and model-based approach for trajectory generation for the mobile manipulator, illustrated in Fig. 1, in a static environment. An OCP considering the nonlinear kinematics of the redundant mobile manipulator, kinematic and dynamic constraints, singularities and collisions, is used to train a NN regression model predicting the optimal solution. Since the NN does not guarantee that the

[1]The authors are with the Institute for System Dynamics, University of Stuttgart, 70563 Stuttgart, Germany {alice.hierholz, andreas.gienger, sawodny}@isys.uni-stuttgart.de

*(Partially) Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2120/1 - 390831618

kinematics of the robot are fulfilled and that the desired TCP target goal is reached, an improvement of the trajectories is carried out using a QP. In the QP, the prediction of the NN is used to formulate the TCP pose as a function of the jacobian matrix at the predicted joint position in order to map the nonlinear kinematics via a linear relationship.

The first main contribution of this work consists of the combination of the data-based and model-based approach for trajectory generation of a redundant mobile manipulator. Pure data-based approaches suffer from a lack of reliability when dealing with small or poorly distributed data sets. The reliability as well as the accuracy is increased significantly by the additional model-based QP. The second main contribution consists of the formulation of the QP using the jacobian matrix at the predicted joint positions, mapping the nonlinear kinematics using a linear relationship. The third main contribution is the reduction of the computational cost when using the combined approach compared to the original OCP.

Below, the OCP used as baseline for the further data-based trajectory generation is illustrated in Section II. It is followed by the regression model implemented as NN in Section III and the QP in Section IV. The results are shown in Section V and a summary and outlook are given in Section VI.

II. OPTIMAL CONTROL PROBLEM

The OCP presented in [12] is used as basis for the data-based trajectory generation in this paper. The most important aspects concerning the system description are therefore briefly summarized here. The OCP is then extended with a singularity consideration. The final formulation of the OCP is then stated.

A. System description

The state-space representation, the forward kinematics and the approximation of the geometry of the used mobile manipulator are presented in the following.

1) State-space representation: The used mobile manipulator, illustrated in Fig. 1, consists of three components. First, an omnidirectional mobile platform whose DOFs are described by two translational joints and one rotational joint. Second, a 6-DOF robotic arm and third, a telescopic axis, also called lift, with 1 DOF which connects the mobile platform to the robotic arm. The mobile manipulator therefore has $n = 10$ DOFs and is thus a redundant system. The states

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{q}} \\ \boldsymbol{x}_{\dot{\boldsymbol{q}}} \end{bmatrix}, \text{ with } \boldsymbol{x}_{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{q}_{\mathrm{pf}} \\ \boldsymbol{q}_{\mathrm{lift}} \\ \boldsymbol{q}_{\mathrm{arm}} \end{bmatrix}, \ \boldsymbol{x}_{\dot{\boldsymbol{q}}} = \begin{bmatrix} \dot{\boldsymbol{q}}_{\mathrm{pf}} \\ \dot{\boldsymbol{q}}_{\mathrm{lift}} \\ \dot{\boldsymbol{q}}_{\mathrm{arm}} \end{bmatrix} \tag{1}$$

hence consist of the joint positions $\boldsymbol{x}_{\boldsymbol{q}} \in \mathbb{R}^n$ and joint velocities $\boldsymbol{x}_{\dot{\boldsymbol{q}}} \in \mathbb{R}^n$. These comprise the generalized coordinates of the mobile platform $\boldsymbol{q}_{\mathrm{pf}} \in \mathbb{R}^3$, the position of the lift $\boldsymbol{q}_{\mathrm{lift}} \in \mathbb{R}$ and the joint positions of the robotic arm $\boldsymbol{q}_{\mathrm{arm}} \in \mathbb{R}^6$ and their respective derivatives $\dot{\boldsymbol{q}}_{\mathrm{pf}}$, $\dot{\boldsymbol{q}}_{\mathrm{lift}}$ and $\dot{\boldsymbol{q}}_{\mathrm{arm}}$. The joint accelerations are the control inputs

$$\boldsymbol{u} = \begin{bmatrix} \ddot{\boldsymbol{q}}_{\mathrm{pf}} \\ \ddot{\boldsymbol{q}}_{\mathrm{lift}} \\ \ddot{\boldsymbol{q}}_{\mathrm{arm}} \end{bmatrix}. \tag{2}$$

Fig. 1: Approximation of the geometry of the mobile manipulator by relevant spheres to prevent self-collisions.

The simplified system equations

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_{\dot{\boldsymbol{q}}} \\ \boldsymbol{u} \end{bmatrix}, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_{t_0} \tag{3}$$

are obtained by neglecting the nonlinear couplings in the dynamic equations. Thus only an integrator chain is considered with the initial states $\boldsymbol{x}_{t_0}$. The validity of this simplification is shown in [12] and is therefore not discussed here.

2) Forward kinematics: In order to incorporate a desired TCP target pose in the OCP, the forward kinematics

$$\boldsymbol{T}_{\mathrm{TCP}} = \mathrm{FK}\left(\mathcal{M}, \boldsymbol{x}_{\boldsymbol{q}}\right) \tag{4}$$

need to be computed. It establishes the connection between given joint positions $\boldsymbol{x}_{\boldsymbol{q}}$, the multi-body model $\mathcal{M}$ of the rigid robot and the homogeneous transformation matrix $\boldsymbol{T}_{\mathrm{TCP}}$.

3) Approximation of the geometry for collision avoidance: Collision avoidance in trajectory generation is essential to generate reasonable trajectories. The method used here is described in detail in [12]. The mobile manipulators geometry is approximated by suitable spheres. Possible collision points are analyzed and based on this only the relevant spheres are kept, which are illustrated in Fig. 1. Then distance constraints are established which describe that the distance between the center of the potentially colliding spheres has to be greater or equal the sum of the radii at all times. These constraints are summarized in the OCP as trajectory constraint $\boldsymbol{g}_{\mathrm{col}}\left(\mathcal{M}, \boldsymbol{x}_{\boldsymbol{q}}(t)\right) \leq \boldsymbol{0}\, \forall t$.

B. Singularity consideration

A singularity analysis of the used robotic arm, a UR10e from Universal Robots, is depicted in [13]. The singularities are divided into shoulder, elbow and wrist singularities. The shoulder singularity occurs when

$$\begin{aligned} f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}\right) = \cos(q_{\mathrm{arm},2})a_2 + \cos(q_{\mathrm{arm},2} + q_{\mathrm{arm},3})a_3 \\ + \sin(q_{\mathrm{arm},2} + q_{\mathrm{arm},3} + q_{\mathrm{arm},4})d_5 = 0 \end{aligned} \tag{5}$$

holds, with the Denavit-Hartenberg parameters from [14]

$$a_2 = -0.6127,\ a_3 = -0.5715,\ d_5 = 0.11985. \tag{6}$$

The elbow singularity occurs for $q_{\mathrm{arm},3} = \{0, \pi, -\pi\}$ and the wrist singularity for $q_{\mathrm{arm},5} = \{0, \pi, -\pi\}$. In order to avoid singularities in the generated trajectories additional constraints depending on the initial joint positions $q_{\mathrm{arm},i}(0)$ with $i = \{2, 3, 4, 5\}$ are introduced. For robustness reasons a parameter $\epsilon > 0$ is introduced to enforce that the trajectories do not get to close to the singularities. The larger ϵ is chosen, the more the robot's workspace is restricted. For avoiding the shoulder singularity it therefore must hold

$$f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t)\right) \begin{cases} \geq \epsilon, & \text{if } f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t_0)\right) > 0 \\ \leq -\epsilon, & \text{if } f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t_0)\right) < 0 \end{cases} \quad \forall t. \tag{7}$$

This condition is enforced in the OCP by means of a trajectory constraint $g_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t)\right) \leq 0$ with

$$\begin{aligned} &g_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t)\right) \\ &= \begin{cases} \epsilon - f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t)\right), & \text{if } f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t_0)\right) > 0 \\ \epsilon + f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t)\right), & \text{if } f_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t_0)\right) < 0 \end{cases} \quad \forall t. \end{aligned} \tag{8}$$

For the elbow and wrist singularity

$$q_{\mathrm{arm},i}(t) \in \begin{cases} \left[-\pi + \epsilon, 0 - \epsilon\right], & \text{if } q_{\mathrm{arm},i}(t_0) \in (-\pi, 0) \\ \left[0 + \epsilon, \pi - \epsilon\right], & \text{if } q_{\mathrm{arm},i}(t_0) \in (0, \pi) \end{cases} \quad \forall t \tag{9}$$

with $i = \{3, 5\}$ must hold. These conditions are directly incorporated into the kinematic box constraints of the joints $\boldsymbol{x}_{\min/\max}$ in the OCP which now depend on the initial joint positions $\boldsymbol{x}_{t_0}$.

C. Trajectory generation

The optimal trajectories for the mobile manipulator with the optimal state variable $\boldsymbol{x}_{\mathrm{opt}}$, the optimal control input $\boldsymbol{u}_{\mathrm{opt}}$ and the optimal end time $t_{\mathrm{f,opt}}$, are generated by the following OCP

$$\min_{\boldsymbol{u}(\cdot)} t_{\mathrm{f}} + \int_{t_0}^{t_{\mathrm{f}}} \boldsymbol{u}^T(t) \boldsymbol{Q} \boldsymbol{u}(t) \mathrm{d}t \tag{10a}$$

$$\text{s.t. } \dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_{\dot{\boldsymbol{q}}} \\ \boldsymbol{u} \end{bmatrix}, \tag{10b}$$

$$\boldsymbol{\Psi}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_{\mathrm{f}})) = \begin{bmatrix} \boldsymbol{x}(t_0) - \boldsymbol{x}_{t_0} \\ \mathrm{FK}\left(\mathcal{M}, \boldsymbol{x}_{\boldsymbol{q}}(t_{\mathrm{f}})\right) - \boldsymbol{T}_{\mathrm{TCP,f}} \\ \boldsymbol{x}_{\dot{\boldsymbol{q}}}(t_{\mathrm{f}}) \end{bmatrix} = \boldsymbol{0}, \tag{10c}$$

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \begin{bmatrix} \boldsymbol{x}(t) - \boldsymbol{x}_{\max}(\boldsymbol{x}_{t_0}) \\ \boldsymbol{x}_{\min}(\boldsymbol{x}_{t_0}) - \boldsymbol{x}(t) \\ \boldsymbol{u}(t) - \boldsymbol{u}_{\max} \\ \boldsymbol{u}_{\min} - \boldsymbol{u}(t) \\ \boldsymbol{g}_{\mathrm{col}}\left(\mathcal{M}, \boldsymbol{x}_{\boldsymbol{q}}(t)\right) \\ g_{\mathrm{sing}}\left(\boldsymbol{x}_{\boldsymbol{q}}(t)\right) \end{bmatrix} \leq \boldsymbol{0}\, \forall t \in \left[t_0, t_{\mathrm{f}}\right], \tag{10d}$$

which is solved using the direct collocation method with piecewise linear states and constant control inputs, implemented in *MATLAB* and *CasADi* [15]. It consists of the cost-functional (10a), the system equations (10b) as in (3), the equality constraints (10c) and the trajectory constraints (10d). The cost-functional consists of the end time t_{f} and a term penalizing the control inputs in order to deal with the redundancy of the system. The weighting factor $\boldsymbol{Q} = \boldsymbol{I}$ is chosen to obtain a trade-off between time-optimality and energy-optimality. The equality constraints enforce the initial and terminating conditions, especially the desired TCP target pose. The trajectory constraints enforce the kinematic constraints of the joints with consideration of the elbow and wrist singularities, the dynamic constraints, the collision avoidance and the consideration of the shoulder singularity.

III. REGRESSION MODEL

A regression model is implemented to predict the optimal trajectories given by the OCP of Section II. Therefore, the process for data generation and the structure of the used NN is described in the following.

A. Data generation

To train, validate, and test the regression model, sufficient data is necessary. Thus the OCP is solved M times for a initial state configuration $\boldsymbol{x}_{t_0}$ and a unique randomly chosen desired TCP target pose

$$\boldsymbol{T}_{\mathrm{TCP,f}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{TCP,f}} & \boldsymbol{p}_{\mathrm{TCP,f}} \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{11}$$

which lies within a certain goal region $\mathcal{G}$. The desired TCP position $\boldsymbol{p}_{\mathrm{TCP,f}} = \begin{bmatrix} x_{\mathrm{TCP,f}} & y_{\mathrm{TCP,f}} & z_{\mathrm{TCP,f}} \end{bmatrix}^T$ is computed by drawing M samples from a uniform distribution for each coordinate. Hence

$$x_{\mathrm{TCP,f}} \sim U(\mathcal{G}_{x_{\min}}, \mathcal{G}_{x_{\max}}), \tag{12a}$$

$$y_{\mathrm{TCP,f}} \sim U(\mathcal{G}_{y_{\min}}, \mathcal{G}_{y_{\max}}), \tag{12b}$$

$$z_{\mathrm{TCP,f}} \sim U(\mathcal{G}_{z_{\min}}, \mathcal{G}_{z_{\max}}), \tag{12c}$$

holds. In order to obtain uniformly distributed desired TCP orientations $\boldsymbol{R}_{\mathrm{TCP,f}}$, unit quaternions are used, as shown in [16]. First three uniformly distributed variables $a_1, a_2, a_3 \sim U(0, 1)$ are generated, from which uniformly distributed unit quaternions are derived by

$$\boldsymbol{q}_{\mathrm{quat}} = \begin{bmatrix} \sqrt{1 - a_1} \sin\left(2\pi a_2\right) \\ \sqrt{1 - a_1} \cos\left(2\pi a_2\right) \\ \sqrt{a_1} \sin\left(2\pi a_3\right) \\ \sqrt{a_1} \cos\left(2\pi a_3\right) \end{bmatrix}. \tag{13}$$

The rotation matrices are then computed from the quaternions as in [17]. Only the desired rotation matrices leading to trajectories which are not ending in singularities are kept. This leads to a final data set of $M^\star < M$ trajectories.

B. Neural network

A fully connected feedforward NN is used as regression model to predict the solution of the OCP. The input consists of the start joint position $\boldsymbol{x}_{\boldsymbol{q}}(t_0)$, the corresponding start TCP pose and the TCP desired target pose. In order to reduce the dimensionality of the input layer, the orientation is described via quaternions, which leads to the input data $\boldsymbol{x}_{\mathrm{in}} \in \mathbb{R}^{24}$. This leads to no additional effort, since quaternions are

already used in the data generation process. The output consists of the optimal joint positions and velocities for each collocation point $k \in [0; N]$, the joint accelerations for $k \in [0; N-1]$ and the optimal end time. The predicted state variables are denoted as $\tilde{\boldsymbol{x}}$, the predicted control inputs as $\tilde{\boldsymbol{u}}$ and the predicted end time as $\tilde{t}_\mathrm{f}$. It therefore holds for the output data $\boldsymbol{x}_\mathrm{out} \in \mathbb{R}^{2n(N+1)+nN+1}$. The Parametric Rectified Linear Unit (PReLU) is used as activation function for all hidden layers. Batch normalization and dropout are applied. Since it is a regression model, the identity is used as activation function of the output layer. Mini batch training using the ADAM optimizer with mean squared error (MSE) loss is used and the data is normalized prior to training. For hyperparameter optimization, the open source hyperparameter optimization framework *Optuna* [18] is used, which implements a Tree-structured Parzen Estimator algorithm. The optimized hyperparameters comprise the number of hidden layers, the layer dimensions and the dropout rate of each layer for the NN architecture and the learning rate, batch size and number of epochs for the optimization. The regression model is implemented in *Python* using *PyTorch* [19] and all training is carried out on a single *NVIDIA RTX A6000*.

IV. QUADRATIC PROGRAMMING

A QP approximating the OCP is used to improve the predicted trajectories, since the regression model does not guarantee that the kinematics of the robot are fulfilled and the desired target pose is reached. A prerequisite for the applicability of the approach is a sufficiently accurate prediction of the NN such that the linearization of the nonlinear kinematics in the QP is valid. This requires a sufficient number of discretization points of the OCP with respect to the end time of the trajectory.

By following the algorithm described in [20] the forward kinematics from (4) are adapted to output unit quaternions instead of rotation matrices. Hence it follows

$$\boldsymbol{p} = \mathrm{FK}_\mathrm{quat}\left(\mathcal{M}, \boldsymbol{x}_{\boldsymbol{q}}\right) \tag{14}$$

with $\boldsymbol{p} \in \mathbb{R}^7$ describing the pose of the TCP with its orientation as quaternion. The differential kinematics

$$\dot{\boldsymbol{p}} = \boldsymbol{J}(\boldsymbol{x}_{\boldsymbol{q}})\boldsymbol{x}_{\dot{\boldsymbol{q}}} \tag{15}$$

are derived from this by derivation with regard to time, with $\dot{\boldsymbol{p}} \in \mathbb{R}^7$ being the translational and rotational velocity of the TCP and $\boldsymbol{J}(\boldsymbol{x}_{\boldsymbol{q}}) \in \mathbb{R}^{7\times n}$ describing the analytical jacobian matrix in quaternion space.

As described in Section III-A, the OCP is solved using the direct collocation method with piecewise linear states and constant control inputs. In order for the QP to best approximate the OCP it therefore must hold for the discretized states

$$\boldsymbol{x}_{\boldsymbol{q}}^{k+1} = \boldsymbol{x}_{\boldsymbol{q}}^{k} + \frac{\delta t}{2}\left(\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k+1} + \boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k}\right) \tag{16}$$

$$\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k+1} = \boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k} + \delta t \boldsymbol{u}^{k} \tag{17}$$

with the time step $\delta t = t^{k+1} - t^k$ and $k \in [0; N-1]$. Analog to this follows the discretized pose of the TCP

$$\boldsymbol{p}^{k+1} = \boldsymbol{p}^{k} + \frac{\delta t}{2}\left(\dot{\boldsymbol{p}}^{k+1} + \dot{\boldsymbol{p}}^{k}\right). \tag{18}$$

Substitution of the differential kinematics from (15) into (18) yields

$$\boldsymbol{p}^{k+1} = \boldsymbol{p}^{k} + \frac{\delta t}{2}\left(\boldsymbol{J}(\boldsymbol{x}_{\boldsymbol{q}}^{k+1})\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k+1} + \boldsymbol{J}(\boldsymbol{x}_{\boldsymbol{q}}^{k})\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k}\right). \tag{19}$$

In order to guarantee unit length of the quaternion in each step k, a normalization must be performed after each step. Since this is not possible in the QP, the normalization is neglected. The results in Section V-B validate the proposed approximation.

Since it is assumed that the prediction from the NN $\left\{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{t}_\mathrm{f}\right\}$ is close to the optimal solution of the OCP $\left\{\boldsymbol{x}_\mathrm{opt}, \boldsymbol{u}_\mathrm{opt}, t_\mathrm{f,opt}\right\}$ and the optimization via the QP produces only small changes in the trajectories, it is further assumed that the linearization of the forward kinematics around $\tilde{\boldsymbol{x}}_{\boldsymbol{q}}$ is valid. Following the same reasoning it is additionally assumed that the time step is approximated by $\tilde{\delta t} = \frac{\tilde{t}_\mathrm{f}}{N}$. Therefore (16) and (17) are reformulated as

$$\boldsymbol{x}_{\boldsymbol{q}}^{k+1} = \boldsymbol{x}_{\boldsymbol{q}}^{k} + \frac{\tilde{\delta t}}{2}\left(\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k+1} + \boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k}\right) \tag{20}$$

$$\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k+1} = \boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k} + \tilde{\delta t}\boldsymbol{u}^{k} \tag{21}$$

and (19) is rewritten as

$$\boldsymbol{p}^{k+1} = \boldsymbol{p}^{k} + \frac{\tilde{\delta t}}{2}\left(\boldsymbol{J}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}^{k+1})\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k+1} + \boldsymbol{J}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}^{k})\boldsymbol{x}_{\dot{\boldsymbol{q}}}^{k}\right) \tag{22}$$

in order to obtain a function linear in the state variables.

With these preliminary considerations, the QP of the form

$$\min_{\bar{\boldsymbol{x}}} \quad \frac{1}{2}\bar{\boldsymbol{x}}^T\boldsymbol{H}\bar{\boldsymbol{x}} + \boldsymbol{f}^T\bar{\boldsymbol{x}} \tag{23a}$$

$$\text{s.t.} \quad \boldsymbol{A}\bar{\boldsymbol{x}} = \boldsymbol{b}, \tag{23b}$$

$$\boldsymbol{lb} \leq \bar{\boldsymbol{x}} \leq \boldsymbol{ub} \tag{23c}$$

can now be set up. It is implemented in *MATLAB* and *CasADi* with the solver *Gurobi* [21]. The optimization variable

$$\bar{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{q}}^{0^T} & \dots & \boldsymbol{x}_{\boldsymbol{q}}^{N^T} & \boldsymbol{x}_{\dot{\boldsymbol{q}}}^{0^T} & \dots & \boldsymbol{x}_{\dot{\boldsymbol{q}}}^{N^T} \\ \boldsymbol{u}^{0^T} & \dots \boldsymbol{u}^{N-1^T} \end{bmatrix}^T \tag{24}$$

with $\bar{\boldsymbol{x}} \in \mathbb{R}^{2n(N+1)+nN}$ consists of the discretized states and control inputs. The prediction of the NN $\{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}\}$ is used as initial value for the optimization variable.

The end time in the cost-functional (10a) must be neglected in the QP. Only the term penalizing the control inputs is adopted in the cost function (23a), which leads to

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & 2\tilde{\delta t}\bar{\boldsymbol{Q}} \end{bmatrix}, \quad \bar{\boldsymbol{Q}} = \begin{bmatrix} \boldsymbol{Q} & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{Q} \end{bmatrix}, \quad \boldsymbol{f} = \boldsymbol{0}, \tag{25}$$

with $\boldsymbol{Q}$ being the weighting factor from the OCP.

The equality constraint (23b) consists of three parts. First, the kinematic equation (20) written in matrix notation

TABLE I: Search space for hyperparameter optimization carried out with *Optuna* and used final hyperparameter set.

	Search space		Final hyperparameter set
	Min	Max	
Hidden layers	5	10	6
Layer dim.	128	512	$\{467, 233, 391, 465, 298, 263\}$
Dropout rate	0.10	0.35	$\{0.31, 0.30, 0.25, 0.16, 0.12, 0.15\}$
Learning rate	1e−4	1e−2	2e−3
Batch size	1024	1280	1057
Epochs	300	450	401

summarized in $\boldsymbol{A}_{11}$, $\boldsymbol{A}_{12}$ and $\boldsymbol{b}_1$. Secondly, the kinematic equation (21) written as $\boldsymbol{A}_{22}$, $\boldsymbol{A}_{23}$ and $\boldsymbol{b}_2$. Lastly, recursive nesting of (22) and reformulation in matrix notation leads to

$$\boldsymbol{A}_{32} = \begin{bmatrix} \boldsymbol{J}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}^0) & 2\boldsymbol{J}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}^1) & \dots & 2\boldsymbol{J}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}^{N-1}) & \boldsymbol{J}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}^N) \end{bmatrix}, \tag{26}$$

$$\boldsymbol{b}_3 = \left[\tfrac{2}{\tilde{\delta t}} \left(\boldsymbol{p}^N - \boldsymbol{p}^0 \right) \right] \tag{27}$$

with the fixed TCP start pose $\boldsymbol{p}^0$ and the desired TCP target pose $\boldsymbol{p}^N$. In summary, this results in the equality constraint

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{22} & \boldsymbol{A}_{23} \\ \boldsymbol{0} & \boldsymbol{A}_{32} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{bmatrix}. \tag{28}$$

The box constraint (23c) enforce the initial and terminating conditions of the joints from (10c) and the state and control box constraints from (10d). The nonlinear collision constraints and shoulder singularity constraints in (10d) must be neglected in the QP. As stated, it is assumed that the prediction from the NN is close to the optimal solution of the OCP and the optimization via the QP produces only small changes in the trajectories, which validates this approximation.

V. RESULTS

The proposed approach is evaluated in the following. First the hyperparameter optimization and the predicted joint space trajectories from the regression model are evaluated. Afterwards follow the results of the improvement of the generated trajectories using the QP and an analysis of the resulting computation times.

A. Trajectory generation by regression model

First, the hyperparameter optimization of the NN is executed to achieve a trade-off between model complexity and accuracy. The data set contains $M^\star = 36000$ trajectories, computed by the OCP with a discretization of $N = 5$. The used search space of the hyperparameters is shown in Tab. I as well as the final hyperparameter set after 1675 trials. For these hyperparameters, the test loss is $3.1\text{e}{-4}$ compared to a training loss of $2.1\text{e}{-4}$. The results imply that no overfitting occurs. The test loss is split into the different predictions for further analysis. All normalized results are summarized in Tab. II, with the key findings highlighted below. The predicted joint position trajectories show a mean MSE of $3.6\text{e}{-4}$, the joint velocities of $4.2\text{e}{-4}$ and the joint accelerations of $1.1\text{e}{-4}$. It is therefore shown that the assumption in Section IV of using the linearization around the predicted joint positions is valid. When looking at the mean MSE of the individual joints it is shown that the two translational joints of the mobile platform have the highest MSE for the position, whereas the lift has the highest MSE for the velocities. When looking at the accelerations, both the translational joints and the lift have the highest MSE. The prediction of the optimal end time has a normalized MSE of $4.3\text{e}{-4} \pm 78\text{e}{-4}$, which proves that the usage of $\tilde{\delta t}$ in the QP is a valid assumption. Upon analysis of the resulting trajectories, it is evident that the predicted trajectories successfully avoid singularities and collisions in all test cases. The mean computation time for the inference of the NN amounts to $490\,\mu\text{s} \pm 20\,\mu\text{s}$.

TABLE II: Evaluation of regression model with normalized mean MSE.

	Position MSE mean±STD e−4	Velocity MSE mean±STD e−4	Acceleration MSE mean±STD e−4
$\boldsymbol{x_q}$	3.6 ± 56	4.2 ± 54	1.1 ± 8.8
$q_{\text{pf},1}$	9.9 ± 131	8.1 ± 95	2.9 ± 27
$q_{\text{pf},2}$	6.5 ± 61	7.1 ± 67	2.3 ± 19
$q_{\text{pf},3}$	4.0 ± 92	6.8 ± 146	1.1 ± 18
q_{lift}	2.3 ± 21	9.9 ± 66	2.9 ± 12
$q_{\text{arm},1}$	3.0 ± 81	3.2 ± 73	0.3 ± 4.0
$q_{\text{arm},2}$	0.3 ± 1.4	0.7 ± 2.9	0.1 ± 0.3
$q_{\text{arm},3}$	1.8 ± 23	0.5 ± 4.7	0.1 ± 0.5
$q_{\text{arm},4}$	1.9 ± 42	1.7 ± 26	0.3 ± 2.2
$q_{\text{arm},5}$	0.7 ± 3.4	0.8 ± 2.9	0.2 ± 0.4
$q_{\text{arm},6}$	5.2 ± 107	3.4 ± 54	0.4 ± 3.8

B. Improvement of trajectories by quadratic programming

The QP is evaluated by using the test data set containing 2400 trajectories. As described in Section IV, the normalization of the quaternion in each step k is neglected. The mean MSE of the normalization error is $1.9\text{e}{-4} \pm 2\text{e}{-4}$, which shows that the approximation is valid. When analysing the resulting trajectories, it is clear that, as for the NN, the computed trajectories are successful in avoiding the neglected shoulder singularity and the collisions in all test cases. The mean optimal value of the cost-functional of the OCP when neglecting the end time is 1.74. The mean optimal cost-function value of the QP is 1.83. It is noticeable that the QP finds a slightly less optimal solution than the OCP, but with a reduction in the mean computing time from $89\,\text{ms} \pm 72\,\text{ms}$ for the OCP to $14\,\text{ms} \pm 71\,\text{ms}$ for the QP in combination with the NN. The QP is able to improve the TCP target reaching accuracy from a mean MSE of $16.3\text{e}{-3} \pm 17.4\text{e}{-3}$ when looking at the trajectory predicted by the NN, to a mean MSE of $6.9\text{e}{-3} \pm 16.1\text{e}{-3}$ for the QP, which represents an improvement by a factor of 2.4. An example of an improved trajectory in task space is illustrated in Fig. 2 in comparison to the solely predicted trajectory of the NN and the optimal trajectory of the OCP. The example scenario shown is deliberately chosen so that the NN performs rather poorly and has high deviations in

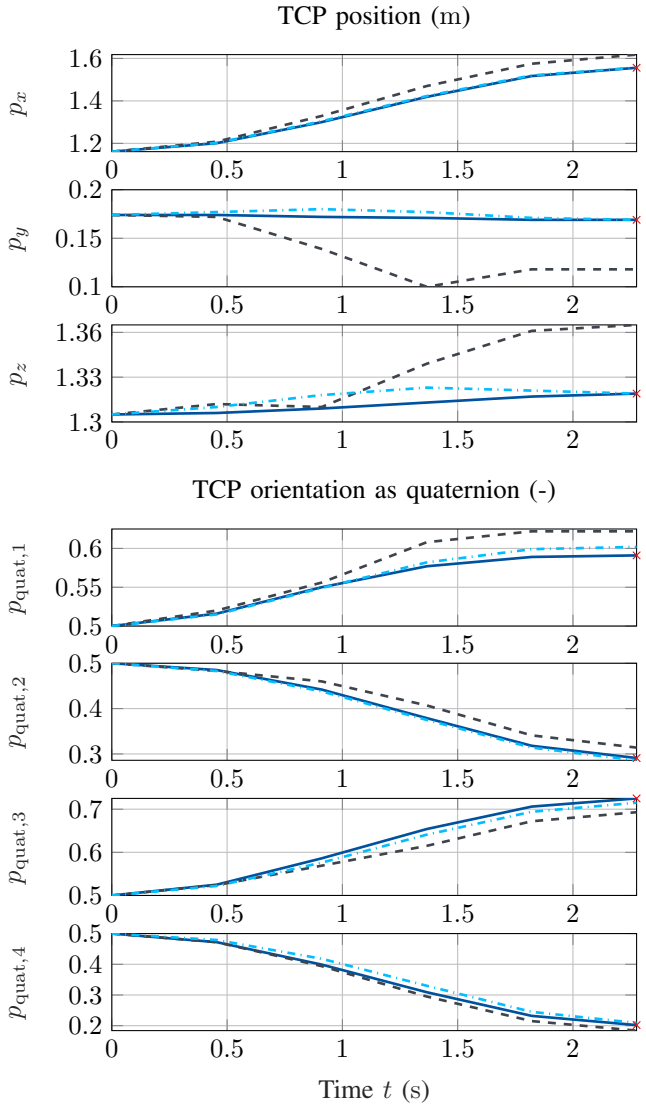

Fig. 2: Generated TCP trajectories through the OCP in (——), the NN in (- - -) and the combination of the NN & QP in (-·-·-) for an exemplary desired TCP goal target (×).

reaching the TCP target, especially for the TCP position. As a result, it is apparent that the QP achieves a significant improvement in the trajectory and the TCP target reaching accuracy.

VI. CONCLUSION

This paper introduced a combination of data- & model-based trajectory generation for redundant mobile manipulators. The regression model implemented as NN shows accurate approximation results of the optimal solution of the OCP considering the system equations, nonlinear kinematic and dynamic constraints, singularities and collisions. The improvement of the predicted trajectories through the presented QP, under the assumption of an sufficiently accurate prediction of the NN and hence a valid linearization, shows low computational cost, compliance with the system equations and an improvement of the TCP target reaching accuracy by a factor of 2.4. In future works, the considered workspace will be extended and different mobile manipulators will be investigated to show general validity of the proposed approach.

REFERENCES

[1] J. Lu, T. Zou, and X. Jiang, "A Neural Network Based Approach to Inverse Kinematics Problem for General Six-Axis Robots," *Sensors*, vol. 22, no. 22, 2022.

[2] T. S. Lembono, E. Pignat, J. Jankowski, and S. Calinon, "Learning Constrained Distributions of Robot Configurations with Generative Adversarial Network," *IEEE Robotics and Automation Letters*, 2021.

[3] G. Jiokou Kouabon, A. Melingui, O. Lakhal, M. Kom, and R. Merzouki, "A Learning Framework to Inverse Kinematics of Redundant Manipulators," *IFAC-PapersOnLine*, vol. 53, no. 2, 2020.

[4] M. Kramer and T. Bertram, "Improving Local Trajectory Optimization by Enhanced Initialization and Global Guidance," *IEEE Access*, vol. 10, 2022.

[5] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion Planning Networks: Bridging the Gap Between Learning-based and Classical Motion Planners," *IEEE Transactions on Robotics*, 2020.

[6] K. Sugiura and H. Matsutani, "P3Net: PointNet-based Path Planning on FPGA," in *2022 International Conference on Field-Programmable Technology (ICFPT)*, 2022.

[7] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural Path Planning: Fixed Time, Near-Optimal Path Generation via Oracle Imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[8] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[9] B. Ichter and M. Pavone, "Robot Motion Planning in Learned Latent Spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, 2019.

[10] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, "A Neural Network Approach for High-Dimensional Optimal Control Applied to Multi-Agent Path Finding," *IEEE Transactions on Control Systems Technology*, 2022.

[11] G. Tang, W. Sun, and K. Hauser, "Learning Trajectories for Real-Time Optimal Control of Quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[12] A. Hierholz, A. Gienger, and O. Sawodny, "Cooperative Time-Optimal Trajectory Generation for a Heterogeneous Group of Redundant Mobile Manipulators," in *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2023.

[13] J. Villalobos, I. Y. Sanchez, and F. Martell, "Singularity Analysis and Complete Methods to Compute the Inverse Kinematics for a 6-DOF UR/TM-Type Robot," *Robotics*, vol. 11, no. 6, 2022.

[14] Universal Robots, "DH Parameters for Calculations of Kinematics and Dynamics," 2024. [Online]. Available: "https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics" (Accessed: 1 Feb 2024).

[15] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi - A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, July 2018.

[16] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*. Springer London, 2009.

[18] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[19] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *33rd Conference on Neural Information Processing Systems*, 2019. [Online]. Available: "https://pytorch.org" (Accessed: 16 Feb 2024).

[20] M. D. Shuster, "A Survey of Attitude Representations," *The Journal of the Astronautical Sciences*, 1993.

[21] Optimization, L. L. C. Gurobi, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: "https://www.gurobi.com" (Accessed: 13 Feb 2024).