Journal of Intelligent & Robotic Systems (2023) 107:57
https://doi.org/10.1007/s10846-023-01848-9

SHORT PAPER

Hierarchical Real-Time Optimal Planning of Collision-Free Trajectories of Collaborative Robots

Dalibor Lukáš[1] · **Tomáš Kot**[1]

Received: 9 June 2022 / Accepted: 3 March 2023 / Published online: 21 April 2023

Abstract
In collaborative robotics the manipulator trajectory has to be planned to avoid collisions, yet in real-time. In this paper we pose the problem as minimization of a quadratic functional among piecewise linear trajectories in the angular (joint) space. The minimization is subjected to novel nonlinear inequality constraints that simplify the original non-penetration constraints to become cheap to evaluate in real time while still preserving collision-avoidance. The very first and most critical step of the computation is to find an initial trajectory that is free of collisions. To that goal we minimize a weighted sum of the violated constraints until they become feasible or a maximal number of steps is reached. Sometimes an incremental growing of the obstacle helps. By incremental growing we mean that we sequentially solve auxiliary subproblems with obstacles growing from ground or falling from top and use as the initial trajectory the one optimized in the previous step. The initial trajectory is then optimized while preserving feasibility at each step. We solve a sequence of simple-bound constrained quadratic programming problems formulated in the dual space of Lagrange multipliers, which are related to the original linearized inequality constraints that are active or close-to-active. Finally, we refine the trajectory parameterization and repeat the optimization, which we refer to as an hierarchical approach, until an overall prescribed time limit, being well below a second, is reached.

Keywords Real-time · Collision-free · Path planning · Hierarchical optimization method · Steepest-descent · Active-set · Inverse kinematics

1 Introduction

Collaborative robotics is an essential topic in the robotic research [1]. A human operator is allowed to support the robotic manipulator with some actions in the working environment [2–4]. The sensoric system, typically depth cameras [5], of the robotic manipulator announces a presence of an obstacle. The control system has to be updated with a new collision-free trajectory. This turns out to be a nontrivial task especially as the new trajectory has to be available in much less than a second, which we refer to as real-time.

There is a lot of algorithmic approaches, but basically the problem is split into two. First of all we need to find a feasible, i.e., collision-free trajectory. Secondly, the trajectory is optimized with respect to energy, for instance. For the former task, which are referred to as planning problems, variants of graph search algorithms are often employed to arrive at a series of way-points [6, 7]. On the other hand optimization problems or nonlinear integrators solve the latter task to follow the way-points in an optimal manner [8–10]. When a real-time collision-free planning [11] is required the graph or discrete planning algorithms become tricky to use especially in case of a high-dimensional configuration space. The main novelty of the present paper is to propose for such higher-dimensional problems a new fast planning algorithm implemented in the C programming language. It relies on tailored collision indicators. Additionally our algorithm is adjusted with two novel strategies: an incremental growing of obstacles and a hierarchical expansion of trajectory control points.

Prior to trajectory planning the inverse kinematics [12, 13] and dynamics [14, 15] of the robotic manipulator is treated. The manipulators may have more degrees of freedom (DOF) than necessarry, which can make the collision-free trajectory planning easier. However, one has to additionally treat singu-

✉ Dalibor Lukáš
dalibor.lukas@vsb.cz

[1] VŠB–Technical University of Ostrava, 17. listopadu 15, 708 00 Ostrava-Poruba, Czech Republic

lar solutions. In case the DOFs are defined in joints the inverse kinematic problem can be seen as finding real-valued roots of a matrix polynomial. Unfortunately, the numerical stability is not guaranteed. When studying such a system, namely the inverse kinematics of the UR3 manipulator, in this paper we develop a geometric insight into the structure of the roots. To our best knowledge these particular derivations have not been published in literature yet.

The rest of the paper is organized as follows: In Section 2 we put our work in the context of existing approaches. In Section 3 we give a novel and computationaly stable geometric derivation of inverse kinematics of the UR3 manipulator. In Section 4 we set the optimal trajectory problem while presenting a novel collision indicators (constraint functions) that are efficient in the sense that the gradient methods can escape from collisions easily and these collision indicators are fast to evaluate at the same time. In Section 5 we present a collision-free planning algorithm relying on the steepest-descent method, a novel incremental growing of obstacles, and a novel hierarchical optimization strategy. We give numerical validations in Section 6 such that for each given starting and goal positions of the effector an optimized collision-free trajectory is found at 273 milliseconds at latest. The paper is concluded in Section 7.

2 Related Work

The first part of our paper presents a novel geometric insight into inverse kinematic solutions of a 6-DOF robotic manipulator UR3. We arrive at new analytic formulae that are numerically stable. In literature the systems comprising of joint DOFs are most frequently solved numerically via finding roots of a matrix polynomial. Many methods rely on the Lyapunov stability theory [16]. In [17] the authors employ the Levenberg-Marquart method. Avoiding self-collisions is treated in [18]. Parameterized classes of inverse kinematic solutions for redundant robots are searched for in [19, 20]. In [21] the inverse kinematics is tailored to a problem of optimal spraying.

The key part of the collision-avoidance system is the optimal trajectory planning algorithm. First of all an initial feasible trajectory has to be found. Among the most frequently used approaches there are variants of graph algorithms such as a D^*-algorithm [22] or an L^*-algorithm [23], which is a linear complexity counterpart to the more traditional A^*-algorithm. Here edges in the graph represent feasible paths between way-points being the graph vertices. These methods are mostly efficient in the 2-dimensional (2D) motions of robotic vehicles. In higher-dimensional configuration spaces such as 6D in our case, the approach is hardly effective in the real-time regime. Another class of methods that is less effective in higher-dimensions rely on rapidly-exploring random trees [24]. On the other hand among the methods that are efficient in higher dimensions there is a class of potential field methods [20, 25–27], in which an artificial potential is introduced such that the trajectory fixed at the start is attracted to the goal and repeled from the obstacles at the same time. In terms of optimization methods [28] this is a penalty approach. A drawback is that it does not always guarantee a collision-free trajectory, in which case the repulsive forces have to be strenghtened. Similar approaches rely on modelling the potential field by the finite element method [29] or by the level-set method [30]. Note that at some specific setups one can develop a geometric approach [31, 32] to the trajectory planning. The above mentioned methods can be combined with stochastic approaches [33, 34] and neural networks [19, 35, 36].

In case the optimization function can be well approximated by a quadratic function the Newton methods [20, 37] become very efficient as they converge with a quadratic rate and they allow for a bigger trajectory changes towards the optimum. The Newton methods are further used successively in the class of sequential convex or quadratic programming algorithms [19, 38, 39]. The tricky part of the algorithms is the treatment of collisions. In [40, 41] the authors develop a class of interior point methods, where the collision constraints are penalized via so-called barrier functions, which suffer from ill-conditioning and also from the fact that the constraints might be slightly violated similarly to the artificial potential methods. In this paper we opt for an active-set approach, where the feasibility of the trajectory is preserved at each iteration. For a survey of the planning algorithms we refer to [42].

To our best knowledge not many papers are devoted to development of efficient collision indicators meaning that their gradients lead to a fast escape from the collision. This is one of the main topics in the present paper. We rely our collision indicators on a fast approximation formulae for the penetration distance and penetration volumes of mutual intersections of cylinders and intersections of cylinders and voxels representing obstacles. A related approach in literature can be found in [22], where the surfaces are covered with point clouds and interactions of their bounding boxes is treated in a hierarchical approach using binary trees. Similarly in [43] the bodies are approximated by series of spheres.

Important task for the trajectory planning is the trajectory parameterization. Having in mind a subsequent optimal path following, smooth trajectories such as Bézier curves [37, 44, 45] are preferable. Here the principal question is how many control parameters we shall take. The more we take the closer to the optimum we get, however, the more computational time it spends. Therefore, hierarchical approaches are benefitial. In [46] information about obstacles is updated using a multi-resolution wavelet basis. In [47] the large optimality systems to be solved is replaced by a hierarchy of

subsystems, where the obstacles are included successively. Similarly, in [48] the successive optimal subproblems alternate among incorporation of single obstacles. In this paper we rely on a hierarchy of trajectory parameterizations. We search for deformations of sub-optimal trajectories from a previous step. Such deformations were presented in [49] without a hierarchical approach.

Though it is not the subject of this paper for the sake of completeness we shall mention some work devoted to a large and important area of optimal path-following. Here an optimal control problem is solved such that a given feasible trajectory is followed in the most efficient way when considering the robot dynamics. One typically aims at a minimum energy consumption and a minimum torque in joints. We refer to the seminal paper [8], where the unconstrained problem is introduced, and to [9], where collisions are taken into account. We refer to [50] for an actual survey.

3 Inverse Kinematics of Robotic Manipulators

3.1 Planar Two-Arm Manipulator

The planar two-arm manipulator is the very fundamental setup, which is worth recalling here. We denote the lengths of arms by a_1 and a_2 and the respective rotational degrees of freedom by $\boldsymbol{\theta} := (\theta_1, \theta_2)$ as depicted in Fig. 1 (left). The positions of arm end-points are denoted by $\boldsymbol{x}^{(1)}$ and $\mathbf{x}^{(2)} := \boldsymbol{x}$. They are computed by the simple means of the following forward kinematics:

$$\boldsymbol{x} := \underbrace{a_1 \begin{pmatrix} \cos\theta_1 \\ \sin\theta_1 \end{pmatrix}}_{=:\boldsymbol{x}^{(1)}} + a_2 \begin{pmatrix} \cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) \end{pmatrix}. \tag{1}$$

For a given position $\boldsymbol{x}$ the inverse kinematics computes the angles θ_1 and θ_2. It relies on the law of cosines, see Fig. 1 (left),

$$\alpha = \arccos\left(\frac{\|\boldsymbol{x}\|^2 + a_1^2 - a_2^2}{2a_1\|\boldsymbol{x}\|}\right),$$
$$\beta = \arccos\left(\frac{a_1^2 + a_2^2 - \|\boldsymbol{x}\|^2}{2a_1a_2}\right). \tag{2}$$

Provided that $\boldsymbol{x}$ is reachable, i.e., $\|\boldsymbol{x}\| \le a_1 + a_2$, we get the so-called lefty (L) and righty (R) solutions

$$\theta_1^{\mathrm{L}} := \gamma + \alpha,\ \theta_2^{\mathrm{L}} := \beta - \pi,\ \theta_1^{\mathrm{R}} := \gamma - \alpha,\ \theta_2^{\mathrm{R}} := \pi - \beta, \tag{3}$$

where $\gamma := \arg(x_1 + \imath x_2)$. Though the inverse kinematics returns 0, 1 (in case $\|\boldsymbol{x}\| = a_1 + a_2$), or 2 solutions, we simplify its notation to $\mathrm{IK2} : \underbrace{\boldsymbol{R}^2}_{\ni \boldsymbol{x}} \to \underbrace{\boldsymbol{R}^2}_{\ni \boldsymbol{\theta}}$, see Fig. 1 (left),

$$\{\boldsymbol{\theta}^{\mathrm{L}}, \boldsymbol{\theta}^{\mathrm{R}}\} := \mathrm{IK2}(\boldsymbol{x}) := \mathrm{IK2}(a_1, a_2, \boldsymbol{x}). \tag{4}$$

3.2 Planar Three-Arm Manipulator

Consider a planar three-arm manipulator of the arm lengths a_1, a_2, and a_3 with the rotational degrees of freedom $\boldsymbol{\theta} := (\theta_1, \theta_2, \theta_3)$, respectively, as depicted in Fig. 1 (right). The kinematics of the arms is denoted by $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$, and $\boldsymbol{x}$, respectively, and it is computed as follows:

$$\mathbf{x} := \underbrace{\underbrace{a_1 \begin{pmatrix} \cos\theta_1 \\ \sin\theta_1 \end{pmatrix}}_{=:\boldsymbol{x}^{(1)}} + a_2 \begin{pmatrix} \cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) \end{pmatrix}}_{=:\boldsymbol{x}^{(2)}} + a_3 \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \tag{5}$$

where $\theta := \theta_1 + \theta_2 + \theta_3$.

Now the inverse kinematics has θ as a parameter, which is actually prescribed as the direction, where the effector should

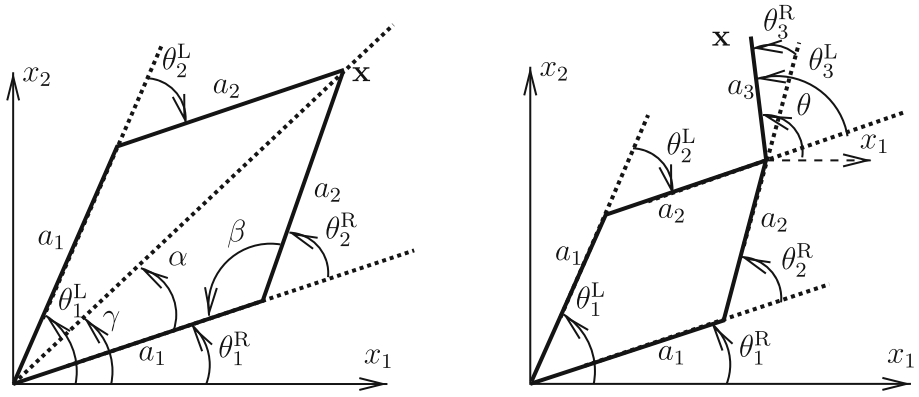

Fig. 1 Planar two-arm (left) and three-arm (right) manipulators

point to. Hence the position and angle of the third arm is given and we arrive back at the inverse kinematics of a two-arm manipulator. As a mapping the inverse kinematics may again return 0 (in case $\|\boldsymbol{x}\| > a_1 + a_2 + a_3$), 1 (in case $\|\boldsymbol{x}\| = a_1 + a_2 + a_3$), or 2 (in case $\|\boldsymbol{x}\| < a_1 + a_2 + a_3$) solutions, which we simplify to IK3 : $\underbrace{\boldsymbol{R}^2}_{\ni \boldsymbol{x}} \times \underbrace{\mathbb{R}}_{\ni \theta} \mapsto \underbrace{\mathbb{R}^3}_{\ni \boldsymbol{\theta}}$.

The evaluation reads as follows:

$$\begin{aligned}\begin{pmatrix}\theta_1^{\mathrm{L/R}}\\ \theta_2^{\mathrm{L/R}}\end{pmatrix} &:= \mathrm{IK2}\left(a_1, a_2, \boldsymbol{x} - a_3\begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix}\right),\\ \theta_3^{\mathrm{L/R}} &:= \theta - \left(\theta_1^{\mathrm{L/R}} + \theta_2^{\mathrm{L/R}}\right).\end{aligned} \tag{6}$$

3.3 Universal Robot UR3

Finally, we consider a 5-arm manipulator with 6 degrees of freedom as depicted in Fig. 2. We denote the arm lengths by $a_i > 0$, $i = 1, 2, \ldots, 5$ and the degrees of freedom by $\theta_1, \theta_2, \ldots, \theta_6$, which are rotations at joints. Additionally, there are three axial displacements $d_2, d_3, d_4 > 0$ of the vertical arms and we denote the total displacement by $d := d_2 - d_3 + d_4$, which is assumed to be positive. At each joint $\iota \in \{1, 2, \ldots, 5\}$ we introduce a local orthogonal coordinate system described by unit vectors $\boldsymbol{i}_\iota, \boldsymbol{j}_\iota, \boldsymbol{k}_\iota \in \mathbb{R}^3$. The kinematics is computed by the following sequence of formulae for the arm end-points and transformations of the local coordinate systems see also Fig. 2 (right):

$$\begin{aligned}
\mathbf{i}_1 &:= \begin{pmatrix}0\\0\\1\end{pmatrix},\ \boldsymbol{j}_1 := \begin{pmatrix}0\\1\\0\end{pmatrix},\ \mathbf{k}_1 := \begin{pmatrix}-1\\0\\0\end{pmatrix},\\
\mathbf{x}_1 &:= a_1\,\mathbf{i}_1,\ (\mathbf{i}_2, \mathbf{j}_2, \mathbf{k}_2) = (\mathbf{i}_1, \mathbf{j}_1, \mathbf{k}_1) \cdot \underbrace{\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\theta_1 & -\sin\theta_1\\ 0 & \sin\theta_1 & \cos\theta_1\end{pmatrix}}_{=:\mathbf{R}_{23}(\theta_1)},\\
\mathbf{x}_2 &:= \underbrace{\mathbf{x}_1 + d_2\,\mathbf{k}_2}_{=:\boldsymbol{x}_{2\mathrm{d}}} + a_2\,(\mathbf{i}_2, \mathbf{j}_2) \cdot \underbrace{\begin{pmatrix}\cos\theta_2\\ \sin\theta_2\end{pmatrix}}_{=:\mathbf{r}(\theta_2)},\ (\mathbf{i}_3, \mathbf{j}_3, \mathbf{k}_3)\\
&= (\mathbf{i}_2, \mathbf{j}_2, \mathbf{k}_2) \cdot \mathbf{R}_{12}(\theta_2),\\
\mathbf{x}_3 &:= \underbrace{\mathbf{x}_2 - d_3\,\mathbf{k}_3}_{=:\boldsymbol{x}_{3\mathrm{d}}} + a_3\,(\mathbf{i}_3, \mathbf{j}_3) \cdot \mathbf{r}(\theta_3),\ (\mathbf{i}_4, \mathbf{j}_4, \mathbf{k}_4)\\
&= (\mathbf{i}_3, \mathbf{j}_3, \mathbf{k}_3) \cdot \mathbf{R}_{12}(\theta_3),\\
\mathbf{x}_4 &:= \underbrace{\mathbf{x}_3 + d_4\,\mathbf{k}_4}_{=:\boldsymbol{x}_{4\mathrm{d}}} + a_4\,(\mathbf{i}_4, \mathbf{j}_4) \cdot \mathbf{r}(\theta_4),\ (\mathbf{i}_5, \mathbf{j}_5, \mathbf{k}_5)\\
&= (\mathbf{i}_4, \mathbf{j}_4, \mathbf{k}_4) \cdot \mathbf{R}_{12}(\theta_4),\\
\mathbf{x} &:= \mathbf{x}_4 + a_5\,(\mathbf{k}_5, -\mathbf{j}_5) \cdot \mathbf{r}(\theta_5).
\end{aligned} \tag{7}$$

Obviously, the last degree of freedom θ_6, the rotation of Joint6, see Fig. 2 (right), does not influence the kinematics unless the robot manipulates an axially nonsymmetric object. Nonetheless, throughout this paper we shall not consider θ_6 any longer. The degrees of freedom of our interest are denoted by $\boldsymbol{\theta} := (\theta_1, \theta_2, \ldots, \theta_5) \in \mathbb{R}^5$.

As for the inverse kinematics, we proceed in three steps. First of all, we resolve the fifth arm and θ_1. Given the effector position $\boldsymbol{x}$ and the unit directional vector $\boldsymbol{v}$ (effector orientation), the fifth arm is determined, $\boldsymbol{x}_4 := \boldsymbol{x} - a_5\,\boldsymbol{v}$. We project $\boldsymbol{x}_4$ onto the plane ρ perpendicular to the direction $\boldsymbol{k}_2 := \boldsymbol{k}_3 := \boldsymbol{k}_4 := \boldsymbol{k}_5 := (-\cos\theta_1, -\sin\theta_1, 0)$ of axial displacements d_2, d_3, d_4, see Fig. 3 (left),

$$\rho := \mathrm{span}\left(\boldsymbol{j}_2 := \begin{pmatrix}-\sin\theta_1\\ \cos\theta_1\\ 0\end{pmatrix}, \boldsymbol{i}_2 := \begin{pmatrix}0\\0\\1\end{pmatrix}\right). \tag{8}$$

The projection reads $\boldsymbol{x}_4^\rho := \boldsymbol{x}_4 - d\,\boldsymbol{k}_2$ and $\boldsymbol{x}_4^\rho \in \rho$ implies the complex-valued equation (rather than the real system) to be solved for η, which is an auxiliary variable and θ_1,

$$\underbrace{(\boldsymbol{x}_4)_1 + \imath\,(\boldsymbol{x}_4)_2}_{=:|\widehat{x}_4|\,\mathrm{e}^{\imath\alpha}} + d\ \underbrace{\mathrm{e}^{\imath\theta_1}}_{\boldsymbol{k}_2} = \eta\ \underbrace{\mathrm{e}^{\imath(\theta_1+\pi/2)}}_{\boldsymbol{j}_2}. \tag{9}$$

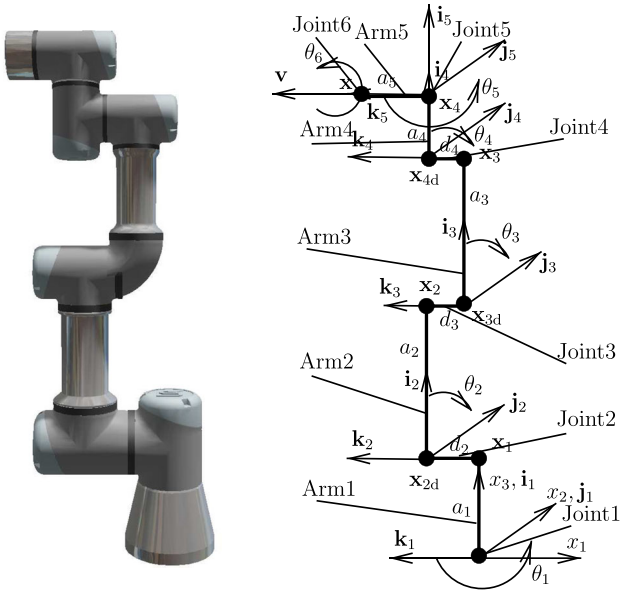

Fig. 2 Universal robot UR3

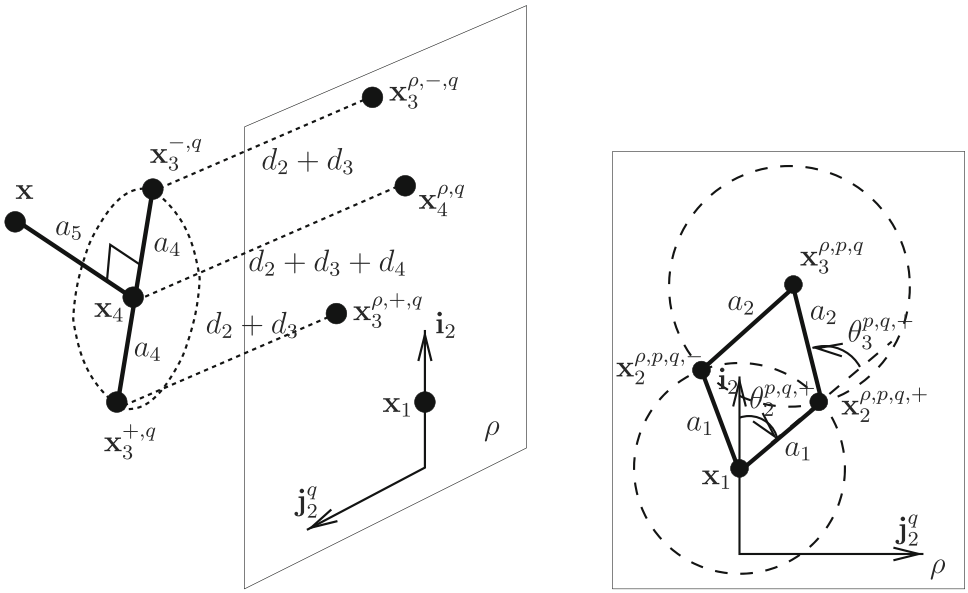

Fig. 3 Determining of the fourth arm of UR3 (left), determining the remaining arms in the plane ρ (right)

Provided $d \le |\widehat{x}_4|$, we get the following two branches of solutions:

$$\theta_1^{\pm} := \alpha \pm \arccos\left(-\frac{d}{|\widehat{x}_4|}\right), \tag{10}$$

where $|\widehat{x}_4| := \sqrt{(\boldsymbol{x}_4)_1^2 + (\boldsymbol{x}_4)_2^2}$ and $\alpha := \arg((\boldsymbol{x}_4)_1 + \imath(\boldsymbol{x}_4)_2)$.

Secondly, we resolve the fourth arm by observing that its direction $\boldsymbol{i}_4 = \boldsymbol{i}_5$ is orthogonal to $\boldsymbol{v}$ as well as to the axial displacement direction $\boldsymbol{k}_2^{\pm}$. The latter implies that $\boldsymbol{i}_4 \in \rho$, which together with $\boldsymbol{v} \perp \boldsymbol{i}_4$ gives the two additional branches of solutions,

$$\boldsymbol{i}_4^{\pm,+} := -(\boldsymbol{v}\cdot\boldsymbol{i}_1)\boldsymbol{k}_2^{\pm} + (\boldsymbol{v}\cdot\boldsymbol{k}_2^{\pm})\boldsymbol{i}_1, \quad \boldsymbol{i}_4^{\pm,-} := -\boldsymbol{i}_4^{\pm,+}, \tag{11}$$

recalling $\boldsymbol{i}_1 := (0,0,1)$. Hence, we get up to four projections of $\boldsymbol{x}_3$ onto ρ,

$$\boldsymbol{x}_3^{\rho,p,q} := \boldsymbol{x}_4^{\rho,p} - a_4\,\boldsymbol{i}_4^{p,q}, \tag{12}$$

where $p, q \in \{+,-\}$. This determines the angle

$$\theta_5^{p,q} := \arg((\boldsymbol{v}\cdot\boldsymbol{k}_5^p) - \imath(\boldsymbol{v}\cdot\boldsymbol{j}_5^{p,q})), \tag{13}$$

where $\boldsymbol{j}_5^{p,q} := \boldsymbol{k}_5^p \times \boldsymbol{i}_5^{p,q}$.

Finally, θ_2 and θ_3 can be determined from the inverse kinematics of the two-arm planar system in ρ, see Fig. 3 (right). We shift the origin to $a_1\boldsymbol{i}_1$ and express $\boldsymbol{x}_3^{\rho,p,q}$ in the coordinate system $\boldsymbol{i}_2 := \boldsymbol{i}_1$ and $\boldsymbol{j}_2^p$,

$$\widetilde{\boldsymbol{x}}_3^{\rho,p,q} := \begin{pmatrix} (\boldsymbol{x}_3^{\rho,p,q} - a_1\,\boldsymbol{i}_1)\cdot\boldsymbol{i}_2 \\ (\boldsymbol{x}_3^{\rho,p,q} - a_1\,\boldsymbol{i}_1)\cdot\boldsymbol{j}_2^p \end{pmatrix}. \tag{14}$$

We have

$$\begin{pmatrix} \theta_2^{p,q,r} \\ \theta_3^{p,q,r} \end{pmatrix} := \mathrm{IK2}(a_2, a_3, \widetilde{\boldsymbol{x}}_3^{\rho,p,q}), \tag{15}$$

where $r \in \{\mathrm{L},\mathrm{R}\}$. The remaining angle can be computed in ρ as follows:

$$\theta_4^{p,q,r} := \arg\left((\boldsymbol{x}_4^{\rho,p} - \boldsymbol{x}_3^{\rho,p,q})\cdot\boldsymbol{i}_4^{p,q,r} + \imath(\boldsymbol{x}_4^{\rho,p} - \boldsymbol{x}_3^{\rho,p,q})\cdot\boldsymbol{j}_4^{p,q,r}\right), \tag{16}$$

where $\boldsymbol{i}_4^{p,q,r} := \cos(\theta_{23}^{p,q,r})\,\boldsymbol{i}_2 + \sin(\theta_{23}^{p,q,r})\,\boldsymbol{j}_2^p$, $\boldsymbol{j}_4^{p,q,r} := -\sin(\theta_{23}^{p,q,r})\,\boldsymbol{i}_2 + \cos(\theta_{23}^{p,q,r})\,\boldsymbol{j}_2^p$, $\theta_{23}^{p,q,r} := \theta_2^{p,q,r} + \theta_3^{p,q,r}$.

To conclude the inverse kinematics of UR3 can be represented as the mapping $\mathrm{IKUR3} : \underbrace{\mathbb{R}^3}_{\ni \boldsymbol{x}} \times \underbrace{\mathbb{R}^3}_{\ni \boldsymbol{v}} \to \underbrace{\mathbb{R}^5}_{\ni \boldsymbol{\theta}}$,

$$\boldsymbol{\theta}^{p,q,r} := \mathrm{IKUR3}(\boldsymbol{x},\boldsymbol{v}) := \mathrm{IKUR3}(a_1,a_2,a_3,a_4,a_5,d,\boldsymbol{x},\boldsymbol{v}), \tag{17}$$

where $\|\boldsymbol{v}\| = 1$, $p, q \in \{+,-\}$, $r \in \{\mathrm{L},\mathrm{R}\}$. In general, there could be either of 0, 1, 2, 4, or 8 solutions.

4 Optimal Collision-Free Trajectory Problem

Given a starting joint position $\boldsymbol{\theta}_{\mathrm{start}}$ and ending position and orientation of the effector, $\boldsymbol{x}_{\mathrm{stop}}$ and $\boldsymbol{v}_{\mathrm{stop}}$, first of all the related ending degrees of freedom are computed by the inverse kinematics $\boldsymbol{\theta}_{\mathrm{stop}} := \mathrm{IKUR3}(\boldsymbol{x}_{\mathrm{stop}}, \boldsymbol{v}_{\mathrm{stop}})$ so that we do not switch between the branches unless necesarry.

The optimal unconstrained trajectory is the straight line $\boldsymbol{\theta}_{\text{line}}(t) := \boldsymbol{\theta}_{\text{start}} + t\,(\boldsymbol{\theta}_{\text{stop}} - \boldsymbol{\theta}_{\text{start}})$, $t \in [0, 1]$. In case of collision we shall search for an as short as possible collision-free curve.

The parameter domain $[0, 1]$ is decomposed into $n + 1$ equidistant intervals, over which continuous piecewise linear trajectories are considered so that the straight line is perturbed in a normal direction as depicted in Fig. 4.

The normal directions are spanned by the 4-dimensional space

$$\operatorname{span} \underbrace{(\boldsymbol{n}_1, \ldots, \boldsymbol{n}_4)}_{=:\boldsymbol{N} \in \mathbb{R}^{5\times 4}} = \operatorname{Null}\left(\boldsymbol{\theta}_{\text{stop}} - \boldsymbol{\theta}_{\text{start}}\right), \tag{18}$$

where $\|\boldsymbol{n}_i\| = 1$. The trajectories pass n break-points and they read

$$\boldsymbol{\theta}(t)(\boldsymbol{p}) := \underbrace{\boldsymbol{\theta}_{\text{start}} + t\,(\boldsymbol{\theta}_{\text{stop}} - \boldsymbol{\theta}_{\text{start}})}_{=\boldsymbol{\theta}_{\text{line}}(t)} + \sum_{i=1}^{n} \varphi_i(t)\, \boldsymbol{N} \cdot \boldsymbol{p}_i, \tag{19}$$

where the vector $\boldsymbol{p} := (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) \in \mathbb{R}^{4n}$ to be optimized describes perturbations of $\boldsymbol{\theta}$ from the unconstrained optimum $\boldsymbol{\theta}_{\text{line}}$ and where $\varphi_i(t)$ are the continuous basis functions that are piecewise linear along the discretization $t_i := \frac{i}{n+1}$, $i = 0, 1, \ldots, n+1$, with the step $h := \frac{1}{n+1}$, i.e., for $i = 1, 2, \ldots, n$,

$$\varphi_i(t) := \begin{cases} \frac{t - t_{i-1}}{h}, & t \in [t_{i-1}, t_i], \\ -\frac{t - t_{i+1}}{h}, & t \in [t_i, t_{i+1}], \\ 0, & \text{elsehere.} \end{cases} \tag{20}$$

We denote the $4n$-dimensional vector space of the trajectories by $\mathcal{T}_n$.

Note that it is rather straightforward to change the non-smooth piecewise linear basis functions to, e.g., cubic splines or to globally smooth basis functions such as Lagrange polynomials. They can be beneficial in terms of preserving a continuous acceleration in joints.

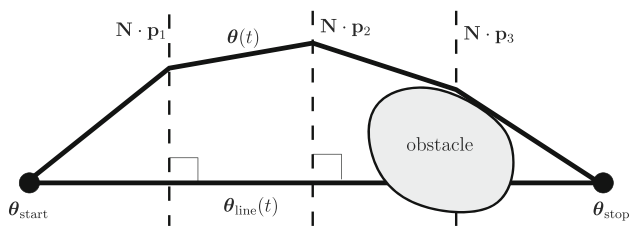

Fig. 4 Piecewise linear parameterization of the trajectory

We solve the following nonlinearly constrained quadratic programming problem:

$$\boldsymbol{p}^* := (\boldsymbol{p}_1^*, \ldots, \boldsymbol{p}_n^*) := \underset{\substack{\boldsymbol{p} \in [-\frac{5}{6}\pi, \frac{5}{6}\pi]^{4n}: \\ \boldsymbol{g}(\boldsymbol{\theta}(t)(\boldsymbol{p})) \le \boldsymbol{0}}}{\arg\min} \|\boldsymbol{p}\|^2, \tag{21}$$

where the bounds $\pm\frac{5}{6}\pi$ are construction limits of joints and where $\boldsymbol{g} : \mathcal{T}_n \to \mathbb{R}^m$ is the constraint function that avoids collisions.

4.1 Constraints Avoiding Collisions

The constraint $\boldsymbol{g}(\boldsymbol{\theta}(t)(\boldsymbol{p})) \le \boldsymbol{0}$ avoids mutual collisions of arms, collisions of arms with a horizontal workspace construction, to which we further refer to as the ground, collisions of arms and joints with a vertical workspace construction, and collisions of arms and joints with voxels representing the obstacle. Namely, referring to Fig. 2, we consider $m := 20$ constraints g_i as follows:

- Constraints $i = 1, 2, 3$ avoid collisions of Arm1 with Arm4, Arm1 with Arm5, and Arm2 with Arm5.
- Constraints $i = 4, 5, 6, 7$ avoid collisions of Arm2, Arm3, Arm4, and Arm5 with the ground $G := \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_3 \le 0\}$.
- Constraints $i = 8, 9, 10, 11$ avoid collisions of Arm3, Joint3, Arm4, and Arm5 with the left quadrant $Q_{\text{L}} := \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 \le -q_1 \text{ and } x_2 \ge q_2\}$, where $q_1, q_2 > 0$ describe positions of two fixed vertical pillars holding the system of cameras. The left pillar is installed at $(-q_1, q_2)$.
- Constraints $i = 12, 13, 14, 15$ avoid collisions of Arm3, Joint3, Arm4, and Arm5 with the right quadrant $Q_{\text{R}} := \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 \ge q_1 \text{ and } x_2 \ge q_2\}$, see the previous item. The right pillar is installed at $\boldsymbol{q} := (q_1, q_2)$, symmetrically (with respect to the plane $x_2 = 0$) to the left pillar.
- Constraints $i = 16, 17, 18, 19, 20$ avoid collisions of Arm2, Arm3, Arm4, Arm5, and Joint3 with voxels representing the obstacle.

Other collisions do not need to be considered due to the dimensions of the universal robot UR3 and the workspace as described in Section 6.

Notice that arms as well as joints are represented by their cylindrical envelops. The joints the collisions of which do not need to be indicated by a collision of connected arms are Joint2, Joint3, and Joint4. They are exactly those related to axial displacements d_2, d_3, d_4 that have not played a role in the inverse kinematics, but they do so now.

To our best knowledge there is no analytic procedure to evaluate collisions of $\boldsymbol{\theta}(t)$ for all $t \in [0, 1]$. Therefore we

discretize the trajectory parameter interval into N equidistant subintervals and consider the constraints at the worst-case discrete parameter as follows:

$$g_i(\boldsymbol{\theta}(t)(\boldsymbol{p})) := \max_{j\in\{0,1,\ldots,N\}} \widetilde{g}_i(\boldsymbol{\theta}(j/N)(\boldsymbol{p})), \tag{22}$$

where $\widetilde{g}_i : \mathbb{R}^5 \to \mathbb{R}$.

4.1.1 Mutual Collisions of Arms

We prescribe the constraint as an approximate depth of the penetration of the two arms as follows:

$$\widetilde{g}_i := r_j + r_k - \mathrm{dist}(S_j, S_k), \tag{23}$$

where S_j, S_k are the axes (segments) and r_j, r_k are the radii of the cylindrical arms. Notice that this approximation of the penetration depth is a strong indicator meaning that it always indicates the true penetration but it can also indicate a false penetration, see Fig. 5 (left). On the other hand it is very cheap to evaluate, which is desired in the real-time optimization. An exact indicator would rely on computing the volume of intersection of the cylinders, which would be computationally demanding.

4.1.2 Collisions of Arms with the Ground

The constraint is prescribed as an approximate depth of the penetration of the arm with the plane $x_3 = 0$ (the ground) as follows:

$$\widetilde{g}_i := r - \min_{t\in[0,1]} \{a_3 + t\,(b_3 - a_3)\}, \tag{24}$$

where a_3, b_3 are the vertical (third) coordinates of the arm end-points and r is the radius. Again, this constraint also gives positive false indications of collisions, see Fig. 5 (right), but it is cheap to evaluate hence proper for the real-time optimization. An exact constraint would evaluate the volume of intersection of the cylinder with the ground, which would be computationally demanding.

4.1.3 Collisions of Arms and Joints with the Quadrants

The constraint approximates the depth of the penetration of an arm or joint with a quadrant. Consider the horizontal projection of the arm/joint axis, $S := \{(x_1, x_2) := (a_1, a_2) + t(b_1 - a_1, b_2 - a_2) \in \mathbb{R}^2 : \ t \in [0, 1]\}$ with the axis end-points $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, the radius r, and the first quadrant $Q := \{(x_1, x_2) \in \mathbb{R}^2 : \ x_1, x_2 \geq 0\}$. We define the level-set function

$$\Phi(S, Q) := \begin{cases} \mathrm{dist}(S, Q), & S \cap Q = \emptyset, \\ -\max\limits_{\boldsymbol{x}\in S\cap Q} \min\{x_1, x_2\}, & S \cap Q \neq \emptyset. \end{cases} \tag{25}$$

The constraint avoiding penetration with the right quadrant $Q_\mathrm{R} := \underbrace{(q_1.q_2)}_{=:\boldsymbol{q}} + Q$ reads

$$\widetilde{g}_i := r - \Phi(S + \boldsymbol{q}, Q), \tag{26}$$

while the one avoiding penetration with the left quadrant $Q_\mathrm{L} := \{(x_1, x_2) \in \mathbb{R}^2 : \ x_1 \leq -q_1, \ x_2 \geq q_2\}$ reads

$$\widetilde{g}_i := r - \Phi(S' + \boldsymbol{q}, Q), \tag{27}$$

where $S' := \{(-x_1, x_2) \in \mathbb{R}^2 : \ (x_1, x_2) \in S\}$. An exact constraint would evaluate the volume of intersection of the cylinder with the quadrants, which would be computationally demanding.

4.1.4 Collisions of Arms and Joints with the Obstacle

This constraint approximates the depth-times-volume of penetration of an arm or joint into the voxels representing the

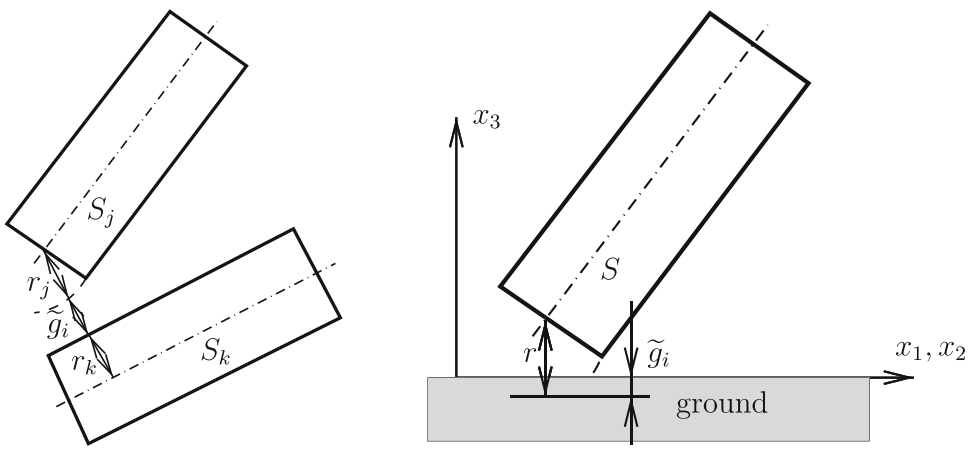

Fig. 5 Mutual collisions of arms (left), collisions of an arm with the ground (right)

obstacle. The product of depth and volume turned out to resolve more collisions than the single depth constraint, like 23 and 24, or the single volume constraint. The voxels are received from cameras and they typically envelop hands of a human operator. Consider a cylindrical arm/joint S determined by radius r and axial end-points $\boldsymbol{a}, \boldsymbol{b}$. Further, consider the union Ω of all the voxels. We compute the constraint in the following four steps:

1. Detect voxels boundary $\Gamma := \partial\Omega$. It collects those square voxel faces that are not shared with another voxel. This step has a quadratic complexity in terms of numbers of voxels, but it is performed only once per evaluation of $\boldsymbol{g}$.
2. Replace each cylinder with a box envelop determined by four edges

$$\boldsymbol{x}^{\pm i}(t) := \boldsymbol{a} \pm \sqrt{2} r\, \boldsymbol{n}^i + t(\boldsymbol{b} - \boldsymbol{a}),\ t \in [0, 1], \tag{28}$$

   where $\boldsymbol{n}^i$, $i = 1, 2$, is an orthonormal basis of $\mathrm{Null}(\boldsymbol{n}^3)$, where $\boldsymbol{n}^3 := (n_1^3, n_2^3, n_3^3) := (\boldsymbol{b} - \boldsymbol{a})/\|\boldsymbol{b} - \boldsymbol{a}\|$. This can be generally discontinuous with respect to $\boldsymbol{n}^3$, but we make it (including all partial derivatives) continuous almost everywhere by the use of spherical coordinates. Namely, we determine the polar ϑ and azimuthal φ angles of $\boldsymbol{n}^3$ i.e. $\vartheta := \arg(n_1^3 + \imath n_2^3)$ and $\varphi := \arccos(n_3^3)$. Then we apply the spherical transformation to $(0, 1, 0)$ and $(0, 0, 1)$ to arrive at $\boldsymbol{n}^1 := \boldsymbol{R}_{13}(\varphi) \cdot \boldsymbol{R}_{12}(\vartheta) \cdot (0, 1, 0)^T$ and $\boldsymbol{n}^2 := \boldsymbol{R}_{13}(\varphi) \cdot \boldsymbol{R}_{12}(\vartheta) \cdot (0, 0, 1)^T$, respectively. Indeed, the resulting mapping $\boldsymbol{n}^3 \mapsto (\boldsymbol{n}^1, \boldsymbol{n}^2)$ is continuous up to the two directions $\boldsymbol{n}^3 = (0, 0, \pm 1)$.
3. Compute intervals of penetrations of the four edges $\boldsymbol{x}^{\pm i}(t)$ into Ω,

$$I^{\pm i} := I_1^{\pm i} \cup \cdots \cup I_{N^{\pm i}}^{\pm i} \subset [0, 1]. \tag{29}$$

   This is done by computing all intersection points of $\boldsymbol{x}^{\pm i}(t)$ with Γ and sorting them along the parameter t. The consecutive pairs of the sorted parameters determine intervals inside, which are of our interest, and outside Ω.
4. Compute the constraint by summing up penetrations of the four edges

$$\widetilde{g}_i := \begin{cases} r^2 \sum\limits_{k \in \{+1,-1,+2,-2\}} \sum\limits_{j=1}^{N^k} |I_j^k| \operatorname{dist}(\boldsymbol{x}^k(t_j^k), \Gamma), & \bigcup\limits_{k \in \{+1,-1,+2,-2\}} I^k \neq \emptyset, \\ r - \operatorname{dist}(S, \Gamma), & \text{elsewhere}, \end{cases} \tag{30}$$

where $|I_j^k|$ is the length of I_j^k and t_j^k is the mid-point of I_j^k.

5 Hierarchical Active-Set Optimization Method

5.1 Finding an Initial Collision-Free Trajectory

The first and most crucial step is to find a feasible trajectory. A natural idea would be to search for a feasible polygonal trajectory in the 5-dimensional space of joint degrees of freedom. However, the global search suffers from an exponential computational complexity. Hence we rely on the following local search algorithm: Given a (zero) non-feasible initial trajectory $\boldsymbol{p}^0$, we proceed the steepest-descent iterations

$$\boldsymbol{p}^{k+1} := \boldsymbol{p}^k + \alpha^k \boldsymbol{d}^k \tag{31}$$

until $\boldsymbol{p}^{k+1}$ is feasible or k, and hence computational time, reaches a given maximum. The descent direction is the negative weighted gradient of non-feasible constraints,

$$\boldsymbol{d}^k := -\sum_{i: g_i(\boldsymbol{p}^k) \geq 0} g_i(\boldsymbol{p}^k) \nabla g_i(\boldsymbol{p}^k), \tag{32}$$

and α^k is computed by the line-search (bisections) method

$$\alpha^k := \operatorname*{arg\,min}_{\substack{\alpha \geq 0: \\ \boldsymbol{p}^k + \alpha \boldsymbol{d}^k \in [-\frac{5}{6}\pi, \frac{5}{6}\pi]}} \max_{i: g_i(\boldsymbol{p}^k) \geq 0} g_i(\boldsymbol{p}^k + \alpha \boldsymbol{d}^k). \tag{33}$$

The bisection starts at the maximal feasible

$$\alpha_0^k := \operatorname*{arg\,max}_{\substack{\alpha \geq 0: \\ \boldsymbol{p}^k + \alpha \boldsymbol{d}^k \in [-\frac{5}{6}\pi, \frac{5}{6}\pi]}} \alpha. \tag{34}$$

If we do not succeed we adopt an incremental growing of the obstacle, represented by the voxels, from the ground as sketched in Fig. 6. First of all we find the maximal vertical coordinate $\overline{x_3} := \arg\max_{\boldsymbol{x} \in \Omega} x_3$. Starting at the unconstrained optimum $\boldsymbol{p} := \boldsymbol{0}$, we solve a sequence of M problems to find a feasible trajectory by algorithm 31, 32, 33, while incrementaly growing the obstacle Ω as follows:

$$\Omega_m := \Omega - \frac{M - m}{M} \overline{x_3} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad m = 1, 2, \ldots, M. \tag{35}$$

The feasible path of the m-th problem is a starting point to the $(m + 1)$-st problem. If we still do not succeed, we let the obstacle fall from the top as follows:

$$\Omega_m := \Omega + \frac{2M - m}{M} \overline{x_3} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad m = M + 1, M + 2, \ldots, 2M. \tag{36}$$

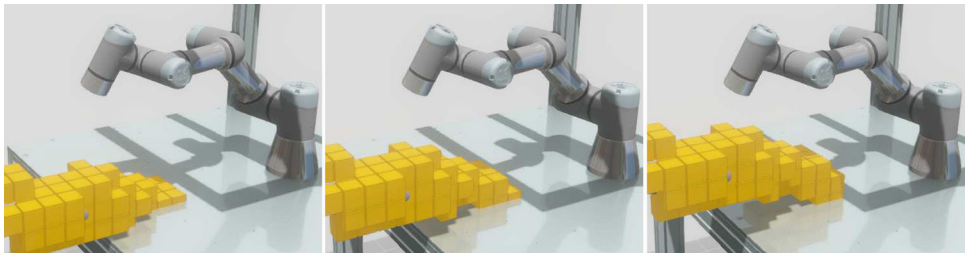

Fig. 6 Incremental growing of the obstacle from the ground

5.2 Optimization

After we find a collision-free trajectory, denoted again by $\boldsymbol{p}^0$, $\boldsymbol{g}(\boldsymbol{p}^0) \leq \boldsymbol{0}$, we finally continue to optimize it. We solve the nonlinearly constrained quadratic programming problem 21 approximately (numerically) by means of the presented optimization algorithms. We employ the sequential quadratic programming and the active-set approach, the iterations of which read

$$\boldsymbol{p}^{k+1} := \boldsymbol{p}^k + \alpha^k\, \boldsymbol{\delta}^k, \quad \alpha^k := \underset{\substack{\alpha\in(0,1]:\\ \boldsymbol{p}^k+\boldsymbol{\delta}^k\in[-\frac{5}{6}\pi,\frac{5}{6}\pi]\\ \boldsymbol{g}(\boldsymbol{p}^k+\alpha\boldsymbol{\delta}^k)\leq\boldsymbol{0}}}{\arg\max\alpha}, \tag{37}$$

where the following quadratic programming subproblem is solved:

$$\boldsymbol{\delta}^k := \underset{\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{p}^k)+(\boldsymbol{G}^k_{\mathcal{A}})^T\cdot\boldsymbol{\delta}=\boldsymbol{0}}{\arg\min} \|\boldsymbol{p}^k+\boldsymbol{\delta}\|^2. \tag{38}$$

The constraint matrix $\boldsymbol{G}^k_{\mathcal{A}}$ comprises of columns $\nabla g_i(\boldsymbol{p}^k)$, $i \in \mathcal{A}$, at active or close-to-active indices. These are chosen at the 10% level of the maximal constraint, i.e.,

$$\mathcal{A} := \left\{ i \in \{1,\ldots,m\} : \; -\frac{1}{10}\max_{j\in\{1,\ldots,m\}}|g_j(\boldsymbol{p}^k)| \leq g_i(\boldsymbol{p}^k) \leq 0 \right\}. \tag{39}$$

Moreover, linearly dependent columns of $\boldsymbol{G}^k_{\mathcal{A}}$ are removed. Hence, 38 is a well-posed linear saddle-point problem that reads as follows:

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{G}^k_{\mathcal{A}} \\ (\boldsymbol{G}^k_{\mathcal{A}})^T & \boldsymbol{0} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\delta}^k \\ \boldsymbol{\lambda}^k \end{pmatrix} = \begin{pmatrix} \boldsymbol{p}^k \\ -\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{p}^k) \end{pmatrix}. \tag{40}$$

The latter admits the solution

$$\boldsymbol{\delta}^k = -\boldsymbol{p}^k - \boldsymbol{G}^k_{\mathcal{A}} \left((\boldsymbol{G}^k_{\mathcal{A}})^T \cdot \boldsymbol{G}^k_{\mathcal{A}} \right)^{-1} \cdot \left(\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{p}^k) - (\boldsymbol{G}^k_{\mathcal{A}})^T \cdot \boldsymbol{p}^k \right). \tag{41}$$

5.3 Hierarchical Optimization

For higher numbers of trajectory break points n, i.e., higher dimensions $4n$, finding a feasible design and its optimization become difficult. Therefore we employ the idea of multigrid methods [51] to solve the problem through a nested hierarchy. At the initial level $l := 1$ we start with $n^{(1)} := 1$ and the unconstrained optimum $\boldsymbol{p}_0^{(1)} := \boldsymbol{0}$. The resulting, so-called triangular, trajectory $\boldsymbol{p}_*^{(1)}$, no matter whether it is feasible or not, is taken as the initial guess $\boldsymbol{p}_0^{(2)}$ at the next level $l := 2$. We introduce two additional break points as depicted in Fig. 7, hence $n^{(2)} := 3$. The process continues at this level and we arrive at the, so-called pentagonal, trajectory $\boldsymbol{p}_*^{(2)}$. The latter is interpolated to the next level $l := 3$, $n^{(3)} := 7$, where we stop. The resulting trajectory is called nonagonal. The interpolations satisfy $\boldsymbol{\theta}^{(l+1)}(t)(\boldsymbol{p}_0^{(l+1)}) = \boldsymbol{\theta}^{(l)}(t)(\boldsymbol{p}_*^{(l)})$. For our choice of basis functions 20 we arrive at the following interpolation matrices $\boldsymbol{P}^{(1,2)}$ and $\boldsymbol{P}^{(2,3)}$:

$$\boldsymbol{p}_0^{(2)} := \underbrace{\left[\begin{pmatrix} 1/2 \\ 1 \\ 1/2 \end{pmatrix} \otimes \boldsymbol{I} \right]}_{=:\boldsymbol{P}^{(1,2)}} \cdot \boldsymbol{p}_*^{(1)},$$

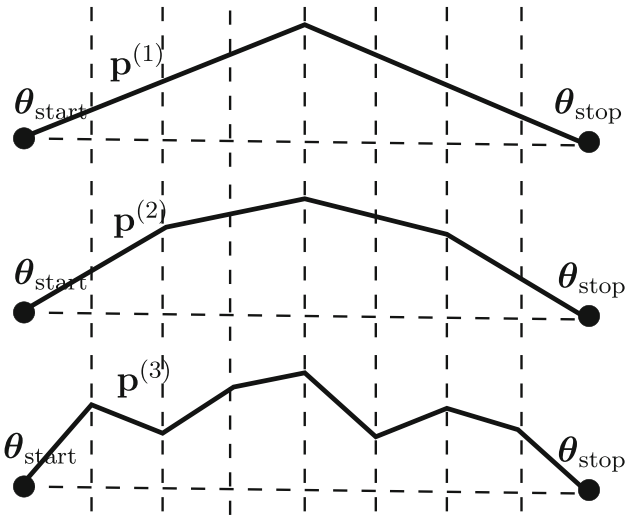

Fig. 7 Hierarchical optimization

$$\boldsymbol{p}_0^{(3)} := \underbrace{\left[\begin{pmatrix} 1/2 & 0 & 0 \\ 1 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \\ 0 & 0 & 1/2 \end{pmatrix} \otimes \boldsymbol{I} \right]}_{=:\boldsymbol{P}^{(2,3)}} \cdot \boldsymbol{p}_*^{(2)} \tag{42}$$

where $\boldsymbol{I}$ is the 4-by-4 identity matrix and $\otimes$ is the Kronecker product meaning that each entry of the matrix on the left is multiplied with $\boldsymbol{I}$. The columns $(\ldots, 1/2, 1, 1/2, \ldots)^T$ are coordinates (shapes) of the basis functions $\varphi_i^{(l)}(t)$ with respect to the next-level basis $(\varphi_j^{(l+1)}(t))_{j=1}^{n^{(l+1)}}$.

6 Numerical Experiments

First of all we shall determine the sizes of the UR3 manipulator and parameters of the optimization problem 21. In the following all the sizes are in meters and the degrees of freedom are in radians. The parameters of UR3 are as follows:

$$a_1 := 0.1585,\ a_2 := 0.24355,\ a_3 := 0.2132,$$
$$a_4 := 0.08535,\ a_5 := 0.0921. \tag{43}$$

The axial displacements of UR3 are

$$d_2 := 0.12,\ d_3 := -0.093,\ d_4 := 0.10405. \tag{44}$$

The upper bound to the radius of all the cylindrical arms and joints is $r := 0.055$. The constraint function 22 is evaluated at $N + 1$ time instances, where $N := 20$. The quadrant Q_{R} is placed at $\boldsymbol{q} := (0.43, 0.2)$.

The steepest-descent optimization algorithm relies on forward numerical derivatives with the step 10^{-6} (radians). The number of steps of the incremental growing is $M := 10$.

We illustrate performance and behaviour of the algorithm on a model situation depicted in Fig. 8.

We consider the following joint coordinates (in radians) of the manipulator:

$$\boldsymbol{\theta}_{\mathrm{start}} := \begin{pmatrix} -0.5297 \\ -1.1799 \\ -0.7909 \\ 0.4001 \\ 1.5708 \end{pmatrix}, \quad \boldsymbol{\theta}_{\mathrm{stop}} := \begin{pmatrix} 0.9521 \\ -1.0796 \\ -1.0071 \\ 0.5160 \\ 1.5708 \end{pmatrix}, \tag{45}$$

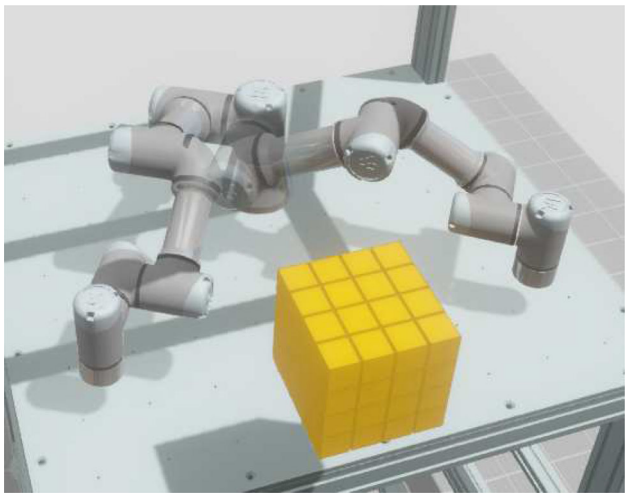

Fig. 8 Starting (left) and ending (right) configurations of UR3 and the cubic obstacle discretized into 64 voxels

where the latter corresponds to the ending position of the effector

$$\boldsymbol{x}_{\mathrm{stop}} := \begin{pmatrix} 0.3195 \\ -0.3884 \\ 0.0694 \end{pmatrix}, \quad \boldsymbol{v}_{\mathrm{stop}} := \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}. \tag{46}$$

The obstacle G is a cube, discretized into $4^3 = 64$ voxels, which is placed at the center at $\boldsymbol{c} := (-0.075, -0.6, 0.325)$. The size of the cube edges is 0.2. The cube is situated parallel to the coordinate system.

In Tab. 1 we give results of numerical simulations for the following 27 shifts of the center of G, where the factor 0.1 [m] is a relative shift of the obstacle that was chosen to simulate as many collisions as possible:

$$\boldsymbol{c} + 0.1\,(i_x, i_y, i_z),\ i_x, i_y, i_z \in \{-1, 0, 1\}. \tag{47}$$

In all these simulation the algorithm found an optimal trajectory unless there was a collision with the obstacle at the ending time instance $t_{\mathrm{stop}} := 1$, which was indicated in less than a millisecond. In some cases the straight line was free of collisions, which was indicated at 1 millisecond. In the other cases typical total computational time was in tens of milliseconds if there was no change of the inverse kinematics branch and below 300 milliseconds if there was a change of the branch.

The first column of Table 1 indicates the shift of the obstacle center. The second column displays whether the optimization succeeded or not and for which inverse kinematic configuration of $\boldsymbol{\theta}_{\mathrm{stop}}$. The configuration 0 denotes the unchanged branch. At some cases the branch had to be changed to the second (nearest to unchanged) one, whichis

Table 1 Results of the numerical experiments

(i_x, i_y, i_z)	solution config.: result	growing steps m	level l	feas. iters.	optim. iters.	total time [ms]
$(-1, -1, -1)$	0: fail	3,3,3/19,19,19				
	1: success	no need	1	1	7	183
$(-1, 0, -1)$	0: success	no need	1	1	3	49
$(-1, 1, -1)$	0: fail			collision at end		0
$(0, -1, -1)$	0: success	no need	1	1	3	46
$(0, 0, -1)$	0: success	no need	1	1	1	38
$(0, 1, -1)$	0: fail			collision at end		0
$(1, -1, -1)$	0: success	no need	1	1	1	37
$(1, 0, -1)$	0: success	10/-	1	3	0	59
$(1, 1, -1)$	0: fail	3,3,3/18,18,18				
	1: success	7,10/-	2	2	3	216
$(-1, -1, 0)$	0: success			straight line		1
$(-1, 0, 0)$	0: fail	2,2,2/20,20,20				
	1: success	6,10/-	2	2	5	192
$(-1, 1, 0)$	0: fail			collision at end		0
$(0, -1, 0)$	0: success			straight line		1
$(0, 0, 0)$	0: success	10/-	1	3	0	45
$(0, 1, 0)$	0: fail			collision at end		0
$(1, -1, 0)$	0: success			straight line		1
$(1, 0, 0)$	0: fail	2,2,2/20,20,20				
	1: success	6,10/-	2	2	4	195
$(1, 1, 0)$	0: fail	2,2,2/20,20,20				
	1: success	5,10/-	2	3	2	273
$(-1, -1, 1)$	0: success			straight line		1
$(-1, 0, 1)$	0: success			straight line		1
$(-1, 1, 1)$	0: fail			collision at end		0
$(0, -1, 1)$	0: success			straight line		1
$(0, 0, 1)$	0: success			straight line		1
$(0, 1, 1)$	0: success			straight line		1
$(1, -1, 1)$	0: success			straight line		1
$(1, 0, 1)$	0: success			straight line		1
$(1, 1, 1)$	0: success			straight line		1

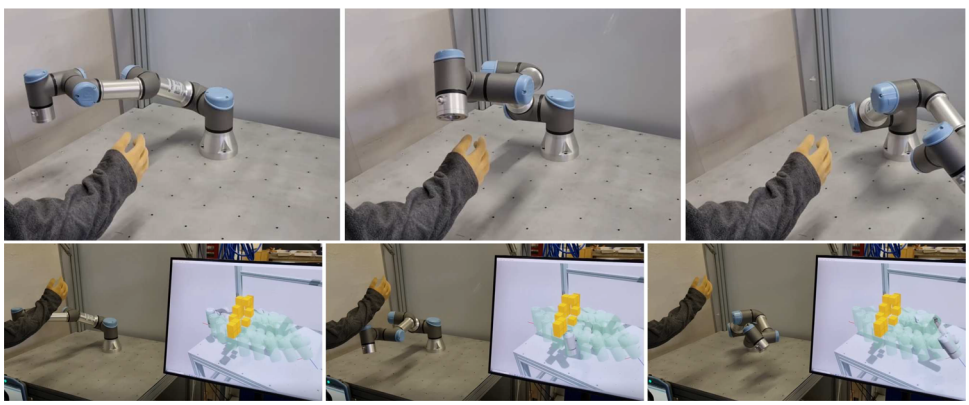

Fig. 9 Examples of a real collision-avoidance in our in-house digital-twin system

denoted by 1. A change to the third branch has never happened. In the third column we depict the number m of growing steps from the ground and from the top. For instance the entry '3,3,3/19,19,19' means that the algorithm failed at step $m := 3$ of the growing from the ground 35 at all the levels $l = 1, 2, 3$ as well as it failed at step $m := 19$ of the growing from the top 36 at all the levels $l = 1, 2, 3$. Recall that $M := 10$. If the algorithm found a feasible path without the growing it is indicated by 'no need'. The entry '7,10/-' indicates that the algorithm failed with the growing from the ground at step $m := 7$ at level $l = 1$, but then it succeeded at level $l = 2$ so that the growing from the top was not needed, which is indicated by that '-'. In the fourth column of Table 1 the level at which we found optimum is displayed. The fifth column displays the number of steps needed to find a feasible trajectory. The sixth column displays the number of the forthcoming steps to find an optimal trajectory. The last column displays the total computational time of the simulation.

Finally, we shall note that the algorithm presented in this paper is a part of a real system including the UR3 manipulator and cameras. In Fig. 9 we sketch a typical human-avoidance situation performed in this real system. The system is implemented in C++. It receives data from a sensory subsystem over Ethernet via UDP (User Datagram Protocol) in the form of a set of voxel centers representing dynamic obstacles. Then the trajectory is updated and the physical UR3 robot is controlled using the RTDE (Real-Time Data Exchange) protocol (Ethernet-based) provided by Universal Robots. Collaborative robots by other manufacturers could be also used, which would require a different communication protocol. 3D visualization is done using our in-house graphical engine based on DirectX 11.1. The camera system relies on an in-house dedicated control unit. It is described in [52].

7 Conclusion

In this paper we have presented a novel approach to optimal planning of collision-free trajectories of robotic manipulators. We arrive at a code that is capable for real-time usage. Namely, the trajectory is found in a quarter of second at latest. Our method relies on

- novel numerically stable formulae derived by geometrical arguments,
- novel fast approximations of collision indicators relying on penetration depth and penetration volume of mutual intersections of cylindrical arms and intersections of cylindrical arms with voxels representing obstacles,
- a novel hierarchical approach within the steepest-descent optimization algorithm.

The ongoing research is now devoted to development of a novel hierarchical optimal path-following method.

Author Contributions Dalibor Lukáš has derived the inverse kinematics, proposed the novel collision-free indicators, carried out the mathematical research, and implemented code of the novel optimization methods. Tomáš Kot has provided robot's specification, software framework for the numerical simulations, and the physical environment in a lab.

Funding Open access publishing supported by the National Technical Library in Prague. This work was funded by the Research Platform focused on Industry 4.0 and Robotics in Ostrava Agglomeration project, project number CZ.02.1.01/0.0/0.0/17_049/0008425 within the Operational Programme Research, Development and Education.

Code Availability Not applicable.

Declarations

Competing Interests The authors have no relevant financial or non-financial interests to disclose.

Ethics Approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Tarbouriech, S., Suleiman, W.: Bi-objective motion planning approach for safe motions: application to a collaborative robot. J Intell Robot Syst **99**, 45–63 (2020)
2. Grushko, S., Vysocký, A., Oščádal, P., Vocetka, M., Novák, P., Bobovský, Z.: Improved mutual understanding for human-robot collaboration: combining human-aware motion planning with haptic feedback devices for communicating planned trajectory. Sensors **21**(11), 3673 (2021)
3. Zhang, S., Li, S., Li, X., Xiong, Y., Xie, Z.: A human-robot dynamic fusion safety algorithm for collaborative operations of cobots. J Intell Robot Syst **104**, 18 (2022)
4. Grushko, S., Vysocký, A., Heczko, D., Bobovský, Z.: Intuitive spatial tactile feedback for better awareness about robot trajectory during human-robot collaboration. Sensors **21**(17), 5748 (2021)

5. Nascimento, H., Mujica, M., Benoussaad, M.: Collision avoidance in human-robot interaction using Kinect vision system combined with robot's model and data. In IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS), pp. 10293—10298 (2020)
6. Latombe, J.-C.: Robot Motion Planning. Springer (1991)
7. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
8. Bobrow, J.E.: Optimal robot path planning using the minimum-time criterion. IEEE J Robot Autom **4**(4), 441–450 (1988)
9. Schiller, Z., Dubowsky, S.: On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. IEEE Trans Robot Autom **7**(6), 785–797 (1991)
10. Lewis, F. L., Dawson, D. M., Abdallah, C. T.: Robot Manipulator Control: Theory and Practice. CRC Press (2003)
11. Kröger, T.: On-Line Trajectory Generation in Robotic Systems. Springer (2010)
12. Bottema, O., Roth B.: Theoretical Kinematics. Dover (1990)
13. Lynch, K.M., Park, F.C.: Modern Robotics. Planning, and Control. Cambridge University Press, Mechanics (2017)
14. Sciavicco, L., Siciliano B.: Modelling and Control of Robot Manipulators. Springer (2000)
15. Tokhi, M.O., Azad, A.K.M.: Flexible Robot Manipulators. Simulation, and Control. The Institution of Engineering and Technology, Modelling (2017)
16. Galicki, M.: Path Following by the End-Effector of a Redundant Manipulator Operating in a Dynamic Environment. IEEE Trans Robotic **20**(6), 1018–1025 (2004)
17. Sugihara, T.: Solvability-Unconcerned Inverse Kinematics by the Levenberg-Marquardt Method. IEEE Trans Robotic **27**(5), 984–991 (2011)
18. Shimizu, M., Kakuya, H., Yoon, W.-K., Kitagaki, K., Kosuge, K.: Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution. IEEE Trans Robotic **24**(5), 1131–1142 (2008)
19. Toshani, H., Farrokhi, M.: Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: A Lyapunov-based approach. Robot and Auton Syst **62**, 766–781 (2014)
20. Safeea, M., Béarée, R., Neto, P.: Collision Avoidance of Redundant Robotic Manipulators Using Newton's Method. J Intell Robot Syst **99**, 673–681 (2020)
21. Kolakowska, E., Smith, S.F., Kristiansen, M.: Constraint optimization model of a scheduling problem for a robotic arm in automatic systems. Robot and Auton Syst **62**, 267–280 (2014)
22. Chen, G., Liu, D., Wang, Y., Jia, Q., Zhang, X.: Path planning method with obstacle avoidance for manipulators in dynamic environment. Int J Adv Robot Syst **15**(6), 1–18 (2018)
23. Niewola, A., Podsedkowski, L.: L* Algorithm – A Linear Computational Complexity Graph Searching Algorithm for Path Planning. J Intell Robot Syst **91**, 425–444 (2018)
24. Bordalba, R., Ros, L., Porta, J.M.: A Randomized Kinodynamic Planner for Closed-Chain Robotic Systems. IEEE Trans Robotic **37**(1), 99–115 (2021)
25. Chen, J.-H., Song, K.-T.: Collision-free motion planning for human-robot collaborative safety under Cartesian constraints, pp. 4348–4354. In IEEE Int Conf on Robotics and Automation, Brisbane (2018)
26. Saeed, R.A., Recupero, D.R., Remagnino, P.: A boundary node method for planning of mobile robots. Robot and Auton Syst **123**, 103320 (2020)
27. Ginesi, M., Meli, D., Roberti, A., Sansonetto, N., Fiorini, P.: Dynamic Movement Primitives: Volumetric Obstacle Avoidance Using Dynamic Potential Functions. J Intell Robot Syst **101**, 79 (2021)
28. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer (2006)
29. Garrido, S., Moreno, L., Blanco, D., Monar, F.M.: Robotic Motion Using Harmonic Functions and Finite Elements. J Intell Robot Syst **59**, 57–73 (2010)
30. Xu, B., Stilwell, D.J., Kurdila, A.J.: Fast Path Re-planning Based on Fast Marching and Level Sets. J Intell Robot Syst **71**, 303–317 (2013)
31. Van Loock, W., Pipeleers, G., Diehl, M., De Schutter, J., Swevers, J.: Optimal Path Following for Differentially Flat Robotic Systems Through a Geometric Problem Formulation. IEEE Trans Robotic **30**(4), 980–985 (2014)
32. Wang, H., Chen, Y., Souères, P.: A Geometric Algorithm to Compute Time-Optimal Trajectories for a Bidirectional Steered Robot. IEEE Trans Robotic **25**(2), 399–413 (2009)
33. Blackmore, L., Ono, M., Williams, B.C.: Chance-Constrained Optimal Path Planning With Obstacles. IEEE Trans Robotic **27**(6), 1080–1094 (2011)
34. Yu, X., Zhou, X., Zhang, Y.: Collision-Free Trajectory Generation and Tracking for UAVs Using Markov Decision Process in a Cluttered Environment. J Intell Robot Syst **93**, 17–32 (2019)
35. Qureshi, A.H., Miao, Y., Simeonov, A., Yip, M.C.: Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners. IEEE Trans Robotic **37**(1), 48–66 (2021)
36. Mac, T.T., Copot, C., Tran, D.T., Keyser, R.D.: Heuristic approaches in robot path planning: A survey. Robot and Auton Syst **86**, 13–28 (2016)
37. Lee, S.-H., Kim, J., Park, F.C., Kim, M., Bobrow, J.E.: Nwton-type algorithms for synamics-based robot movement optimization. IEEE Trans Robotic **21**(4), 657–667 (2005)
38. Mirolo, C., Carpin, S., Pagello, E.: Incremental Convex Minimization for Computing Collision Translations of Convex Polyhedra. IEEE Trans Robotic **23**(3), 403–415 (2007)
39. Zips, P., Böck, M., Kugi, A.: Optimisation based path planning for car parking in narrow environments. Robot and Auton Syst **79**, 1–11 (2016)
40. Alba, M., Ribieiro, L., Herskovits, J.: Trajectory Optimization of Industrial Robots with a Feasible Direction Interior Point Algorithm. In Proceedings of the 6th International Conference on Engineering Optimization, pp. 1360—1371. Springer (2019)
41. Herskovits, J.: Feasible Direction Interior-Point Technique for Nonlinear Optimization. J Optim Theory Appl **99**(1), 121–1476 (1998)
42. Mohaman, M.G., Salgoankar, A.: A survey of robotic motion planning in dynamic environments. Robot and Auton Syst **100**, 171–185 (2018)
43. Zhao, L., Zhao, J., Liu, H.: Solving the Inverse Kinematics Problem of Multiple Redundant Manipulators with Collision Avoidance in Dynamic Environments. J Intell Robot Syst **101**, 30 (2021)
44. Yang, K., Sukkarieh, S.: An Analytical Continuous-Curvature Path-Smoothing Algorithm. IEEE Trans Robotic **26**(3), 561–568 (2010)
45. Klančar, G., Škrjanc, I.: A Case Study of the Collision-Avoidance Problem Based on Bernstein-Bézier Path Tracking for Multiple Robots with Known Constraints. J Intell Robot Syst **60**, 317–337 (2010)
46. Tsiotras, P., Jung, D., Bakolas, E.: Multiresolution Hierarchical Path-Planning for Small UAVs Using Wavelet Decompositions. J Intell Robot Syst **66**, 505–522 (2012)
47. Nizar, I., Jaafar, A., Hidila, Z., Barki, M., Illoussamen, E.H., Mestari, M.: Effective and Safe Trajectory Planning for an Autonomous UAV Using a Decomposition-Coordination Method. J Intell Robot Syst **103**, 50 (2021)
48. Moghaddam, S.K., Masehian, E.: Planning Robot Navigation among Movable Obstacles (NAMO) through a Recursive Approach. J Intell Robot Syst **83**, 603–634 (2016)

49. Pham, Q.-C., Nakamura, Y.: A New Trajectory Deformation Algorithm Based on Affine Transformations. IEEE Trans Robotic **31**(4), 1054–1063 (2015)
50. Rubi, B., Pérez, R., Morcego, B.: A Survey of Path Following Control Strategies for UAVs Focused on Quadrotors. J Intell Robot Syst **98**, 241–265 (2020)
51. Brenner, S.C., Scott, L.R.: The Mathematical Theory of Finite Element Methods. Springer (2008)
52. Oščádal, P., Spurný, T., Kot, T., Grushko, S., Suder, J., Heczko, D., Novák, P., Bobovský, Z.: Distributed camera subsystem for obstacle detection. Sensors **22**(12), 4588 (2022)

Dalibor Lukáš received his M.S. and Ph.D. degree in applied mathematics and computer science from VŠB-Technical University of Ostrava in 1999 and 2003, respectively. His dissertation was on optimal shape design in magnetostatics. In 2000–2004 he was employed in a Special Research Programme at Johannes Kepler University in Linz, Austria. In 2011–2016 he was a vice-head of a research programme at the National Supercomputing Center IT4Innovations at VŠB-Technical University of Ostrava, Czech Republic. He is currently an Associated Professor at the Department of Applied Mathematics at VŠB-Technical University of Ostrava. There he is the chair of the doctoral study programme Computational and Applied Mathematics and he is the head of the research group Numerical Analysis and HPC. His research focus in applied mathematics and code development is driven by industrial applications the simulations of which are behind the scope of current commercial codes. He has co-investigated projects on electromagnetic forming of metalic sheets, on structural health monitoring in aeronautics, on time-reversal nondestructive testing, and currently on collaborative robotics.

Tomáš Kot received the M.Sc. and Ph.D. degrees in robotics from the Technical University of Ostrava, Czech Republic, in 2004 and 2011, respectively.

In 2020, he finished his habilitation at the same university, where he currently works as a Senior Researcher. His research activities focus on complex simulations and control of mechatronic systems, visualisation, application of virtual and augmented reality in robotics, optimisation of layouts of robotised workplaces, algorithms for automatic design of an optimal kinematic structure of a robotic manipulator suitable for a given task, and lately also collision avoidance for collaborative robots sharing workspace with human workers.