Kinematic Trajectory Planning for Dynamically Unconstrained Nonprehensile Joints

Markus M. Schill and Martin Buss

***Abstract*—This letter considers an augmented kinematic formulation for nonprehensile manipulation through intermittent contacts as occurring in catching, batting, or juggling. In such scenarios, the contact point with an end-effector is variable, which we propose to model with additional virtual joints at the end of the kinematic chain. While not in contact with the manipulated part, these new joints are unconstrained in terms of velocity and acceleration. An optimization based and, thus, tuning-free comparison of differential inverse kinematic approaches is carried out, given path or trajectory of the manipulation task is known. Simulations and an experiment show that the proposed augmentation enables dynamically feasible acceleration variations at high velocities on and close to a given path.**

***Index Terms*—Dexterous manipulation, redundant robots, motion and path planning.**

I. Introduction

REPEATABLE success of robotic manipulation depends on robust solutions in perception, modeling and control. Robustness taken by itself, however, is only a necessary criterion for the practical applicability of individual manipulation solutions (also referred to as *primitives*). Only in combination with efficiency (e.g., throughput, range and cost) gaps between academic and commercial interest may be bridged. As an example, throwing and catching parts has potential to increase the efficiency of industrial object transport. Nonetheless, published work regarding this problem has so far been forced to make trade-offs between robustness and efficiency when it comes to experiments.

Given a manipulation primitive candidate, typically formulated in Cartesian task space, experiments with redundant manipulators are conducted using one out of many available inverse kinematic algorithms. Within this step from simulation to experiment, constraints on limits, velocities, accelerations, and torques of the manipulator joints must be met to maintain robustness claims made in task space. In the object transport example, this last step becomes challenging when a robot manipulator should catch a fast flying part. Solutions to the catching problem have recently advanced from static interception of the flight trajectory [1], [2] to an adaptation of the end-effector velocity [3]–[5]. For provably robust catching, the manipulator end-effector might even have to follow the flight trajectory for some time with a particular acceleration pattern relative to the part. So far, these adapting approaches have to reduce the end-effector velocity leading to impacts with the part [3], [5], which may cause damage. Alternatively, redundancy is resolved offline [4]. A key to further improvement is thus the ability to solve the inverse kinematic problem in real-time while operating close to, or even beyond, classical manipulator constraints.

The field of nonprehensile dynamic manipulation [6], [7] is well suited to tackle insufficient manipulator dynamics. Unlike grasping approaches [2], a generic nonprehensile end-effector provides a large potential contact area. This area can augment the kinematics by additional virtual prismatic joints for those manipulation primitives, which are based on impacts or transitions into continuous contact. Most importantly, these virtual joints are unconstrained with respect to velocity or acceleration. After a transition into continuous contact as in catching, e.g., work on nonprehensile rolling manipulation [8], [9] becomes applicable. Further examples for manipulation with intermittent contacts besides catching are e.g., juggling [10]–[12] and batting [13], [14].

A body of work exists to resolve redundancies with joint constraints including limitations on velocity and acceleration. Most common is the use of a weighted pseudo-inverse that locally minimizes joint velocities in real-time. In the presence of joint displacement limits, the weights are usually formulated joint dependent [15] or the gradient of a cost function is projected into the null-space of the inverse kinematic solution [16]. When a joint still gets close to a limit, the pseudo-inverse approaches cannot guarantee accurate task execution anymore, which can be countered by iterative joint velocity saturation [17]. Also practical to accurately track a path is the dynamic scaling of joint trajectories [18] or the task [19]. If the task is a priori known, optimal control in combination with trajectory deformation is applicable [20] but does not allow for real-time acceleration changes anymore.

This letter contributes to the field of nonprehensile dynamic manipulation as follows. (1) The notion of additional, dynamically unconstrained joints is introduced. These virtual joints become available with nonprehensile dynamic manipulation through intermittent contacts. (2) The redundancies gained with the new joints are exploited to enable fast end-effector motions on (or close to) a known path with real-time acceleration input.

Manuscript received July 29, 2017; accepted December 16, 2017. Date of publication December 29, 2017; date of current version January 16, 2018. This paper was recommended for publication by Associate Editor K. Harada and Editor T. Asfour upon evaluation of the reviewers' comments. This work was supported by the ERC Advanced Grant SHRINE Agreement no. 267877. *(Corresponding author: Markus M. Schill.)*

The authors are with the Chair of Automatic Control Engineering and TUM Institute for Advanced Study, Technical University of Munich, München 80333, Germany (e-mail: m.schill@tum.de; mb@tum.de).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LRA.2017.2788197

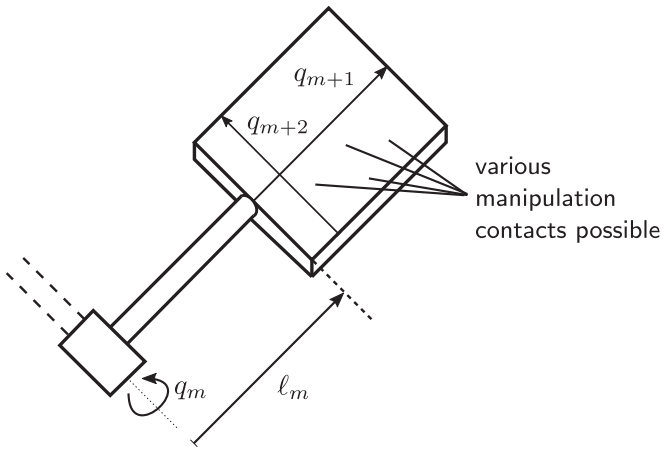

Fig. 1. In nonprehensile dynamic manipulation, a generic planar end-effector (e.g., [3], [4]) at the end of the kinematic chain increases redundancy. For manipulation with intermittent contacts, such as catching or batting, these additional virtual prismatic joints provide unlimited velocity and acceleration potential.

(3) Inverse differential kinematic trajectory planners are compared unbiased and tuning-free based on weight optimization.

Particular attention from formulation until experimental evaluation is owed to the infinite velocity and acceleration potential of the new virtual joint notion. A planar catching experiment reported at the end of this letter illustrates how the new joints enable fast, accurate and still flexible task execution with a formerly non-redundant robot.

II. Dynamically Unconstrained Nonprehensile Joints

This section first introduces the extension of classical kinematics with virtual nonprehensile prismatic joints for manipulation through intermittent contacts. Then, constraints are formulated, which must hold to guarantee that tasks perform accurately in experiments.

A. Augmented Kinematics With Unconstrained Joints

Let an n-dimensional manipulation task $\boldsymbol{x} \in \mathbb{R}^n$ be solved with a manipulator that has m degrees of freedom (DOF) denoted by $\boldsymbol{q}_\mathrm{m} \in \mathbb{R}^m$. Furthermore, assume the manipulation task requires part and end-effector to come into contact in a (temporarily) nonprehensile way. Then, there mostly exists an area, instead of a single point, on the end-effector that is eligible for the contact, cf. Fig. 1. This circumstance may be reflected if $\boldsymbol{q}_\mathrm{m}$ is augmented with a additional (prismatic) joints denoted by $\boldsymbol{q}_\mathrm{a} \in \mathbb{R}^a$, which leads to $\boldsymbol{q} = \left[\boldsymbol{q}_\mathrm{m}^T \boldsymbol{q}_\mathrm{a}^T\right]^T$. These augmented joints are typically located at the end of the kinematic chain and are limited to $a \in \{1, 2\}$.

In terms of joint constraints, the m traditional joints $\boldsymbol{q}_\mathrm{m}$ and the a augmented, virtual joints $\boldsymbol{q}_\mathrm{a}$ show, however, major differences. While not in contact with the manipulated part, the augmented joints provide – in a kinematic sense – infinite velocity and acceleration capabilities. Around the predicted contact time, velocity equality constraints may be employed to control the tangential friction effect at collision. Alternatively, the manipulation task is robust to tangential disturbances [4] and thus unconstrained joint velocity applies throughout the task. The latter is assumed for the remainder of this letter.

B. Joint Constraints

Accurate tracking of a task $\boldsymbol{x}(t)$ with a manipulator is only possible if the joint space trajectory stays within the manipulator constraints

$$\underline{\boldsymbol{Q}} \leq \boldsymbol{q} \leq \overline{\boldsymbol{Q}} \quad \text{(joint ranges)} \tag{1a}$$

$$\underline{\boldsymbol{V}} \leq \dot{\boldsymbol{q}}_\mathrm{m} \leq \overline{\boldsymbol{V}} \quad \text{(velocity limits)} \tag{1b}$$

$$\underline{\boldsymbol{T}} \leq \boldsymbol{\tau} \leq \overline{\boldsymbol{T}} \quad \text{(torque limits).} \tag{1c}$$

Recalling the unconstrained velocity potential of the augmented joints introduced in Section II-A, the velocity limits (1b) and torque limits (1c) are thus of dimension m, whereas the joint limits (1a) are of dimension $(m + a)$. The torque limits (1c) relate to acceleration limits using the dynamic equation

$$\boldsymbol{M}(\boldsymbol{q}_\mathrm{m})\ddot{\boldsymbol{q}}_\mathrm{m} + \boldsymbol{C}(\boldsymbol{q}_\mathrm{m}, \dot{\boldsymbol{q}}_\mathrm{m})\dot{\boldsymbol{q}}_\mathrm{m} + \boldsymbol{G}(\boldsymbol{q}_\mathrm{m}) = \boldsymbol{\tau}\ , \tag{2}$$

with the inertia matrix $\boldsymbol{M} \in \mathbb{R}^{m\times m}$, the Coriolis matrix $\boldsymbol{C} \in \mathbb{R}^{m\times m}$, the gravitational vector $\boldsymbol{G} \in \mathbb{R}^m$, and the input torque $\boldsymbol{\tau} \in \mathbb{R}^m$.

The constraints (1) are collected in a generalized constraint vector $\boldsymbol{h} \in \mathbb{R}^N$, where $\boldsymbol{h} = \left[\boldsymbol{q}^T\ \dot{\boldsymbol{q}}_\mathrm{m}^T\ \boldsymbol{\tau}^T\right]^T$, and thus $N = 3m + a$ in the above case. As a result, the constraints (1) become briefly

$$\underline{\boldsymbol{h}} \leq \boldsymbol{h}(t) \leq \overline{\boldsymbol{h}}, \tag{3}$$

where the lower and upper bounds are denoted $\underline{\boldsymbol{h}}^T = \left[\underline{\boldsymbol{Q}}^T\ \underline{\boldsymbol{V}}^T\ \underline{\boldsymbol{T}}^T\right]$ and $\overline{\boldsymbol{h}}^T = \left[\overline{\boldsymbol{Q}}^T\ \overline{\boldsymbol{V}}^T\ \overline{\boldsymbol{T}}^T\right]$, respectively. The constraint definition (3) can be extended to meet additional limitations such as motor power limits using $h_j = \tau_j \dot{q}_j$ with $j = 1, \ldots, m$.

III. Optimized Redundancy Resolution

This section recalls weighting-based inverse differential kinematic planners to resolve redundancies in real-time while staying within the introduced constraints. Given a nominal task trajectory, off-line optimization of the weights and the initial augmented state then minimizes the distance to constraints over time.

A. Redundancy Resolution

Let $\boldsymbol{f} : \mathbb{R}^{m+a} \to \mathbb{R}^n$ describe the map from joint space to task space. Then, given a manipulator configuration $\boldsymbol{q}$, the $n \times (m + a)$ Jacobian is defined as $\boldsymbol{J}(\boldsymbol{q}) = \partial \boldsymbol{f}(\boldsymbol{q})/\partial \boldsymbol{q}$ and the differential kinematic equation is

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}. \tag{4}$$

In order to obtain the desired joint velocities for a given task, the differential kinematics (4) must be inverted. In case of $(m + a) > n$ the inversion problem is redundant and thus infinitely many solutions exist. A common way to resolve the

redundancy in real-time is the use of a pseudo-inverse

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{W}} \dot{\boldsymbol{x}} \tag{5}$$

with

$$\boldsymbol{J}_{\boldsymbol{W}} = \boldsymbol{W}^{-1} \boldsymbol{J}^T \left(\boldsymbol{J} \boldsymbol{W}^{-1} \boldsymbol{J}^T \right)^{-1}, \tag{6}$$

which locally minimizes the joint velocities according to a symmetric positive definite weighting matrix $\boldsymbol{W} \in \mathbb{R}^{(m+a)\times(m+a)}$. Additional objectives, e.g., respecting the limited range of the augmented joints, are included by extending (5) to

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{W}} \dot{\boldsymbol{x}} + k \left(\boldsymbol{I}_{m+a} - \boldsymbol{J}_{\boldsymbol{W}} \boldsymbol{J} \right) \nabla w(\boldsymbol{q}), \tag{7}$$

which optimizes a secondary objective function $w(\boldsymbol{q})$ by projecting its gradient $\nabla w = \partial w(\boldsymbol{q}) / \partial \boldsymbol{q}$ into the null space of the Jacobian. The factor $k \in \mathbb{R}$ weights the secondary task in relation to the primary, velocity minimizing, task, whereas positive or negative signs result in maximization or minimization, respectively. Note that the joint trajectory planners (5) and (7) lead to different solutions for every possible initial configuration $\boldsymbol{q}_0$. Hence, the choice of $\boldsymbol{q}_0$ is at least as important as the tuning of $\boldsymbol{W}$ and k to find feasible joint trajectories through the entire task. In practice, such tuning often turns out unsuitable and unintuitive in the presence of acceleration or torque constraints. Therefore, we formulate a constrained optimization problem in the next section to find $\boldsymbol{W}$, k and $\boldsymbol{q}_0$ automatically based on a nominal task trajectory.

B. Constrained Dynamic Optimization Problem

Given a nominal task trajectory $\boldsymbol{x}^*(t)$ with $t \in [t_0, t_\mathrm{f}]$, an initial joint configuration $\boldsymbol{q}_\mathrm{m}(t_0)$, and an initial guess for $\boldsymbol{q}_\mathrm{a}(t_0)$, $\boldsymbol{W}$ and k, the desired joint trajectory follows from numeric differentiation and integration of (5) or (7). Hence, defining a vector of optimization variables

$$\boldsymbol{c} = \left[\boldsymbol{q}_\mathrm{a}^T(t_0) \ \mathrm{diag}\left(\boldsymbol{W}\right)^T \ k \right]^T, \tag{8}$$

the constraint vector in (3) becomes $\boldsymbol{h}(\boldsymbol{x}^*(t), \boldsymbol{c})$, where we choose $\boldsymbol{W}$ as a diagonal matrix. Note here that other task parameters can easily be added to $\boldsymbol{c}$, e.g., the starting point $\boldsymbol{x}(t_0)$ in the manipulator workspace.

In order to ensure that the joint trajectory not only stays within the constraints, but also keeps a distance to each constraint, an inverse distance function

$$H_i\left(h_i(\boldsymbol{x}^*(t), \boldsymbol{c})\right) = \frac{\left(\bar{h}_i - h_i\right)^2}{\left(\bar{h}_i - h_i\right)\left(h_i - \underline{h}_i\right)} \tag{9}$$

similar to [21] is used. Note that with (9) the value H_i becomes infinite when the i-th element of the vector $\boldsymbol{h}$ reaches one of its constraints. Further constraint violations (i.e. $h_i < \underline{h}_i$ or $h_i > \bar{h}_i$) result in (9) becoming negative. With the help of (9) a dynamic optimization problem is formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{c}}{\text{minimize}} \ & \int_{t_0}^{t_f} \sum_{i=1}^{N} H_i\left(h_i(\boldsymbol{x}^*(t), \boldsymbol{c})\right) \mathrm{d}t \\ \text{s.t. } & H_i \geq 0 \ \forall \ i \in \{1, 2, \ldots, N\}, t \in [t_0, t_f]. \end{aligned} \tag{10}$$

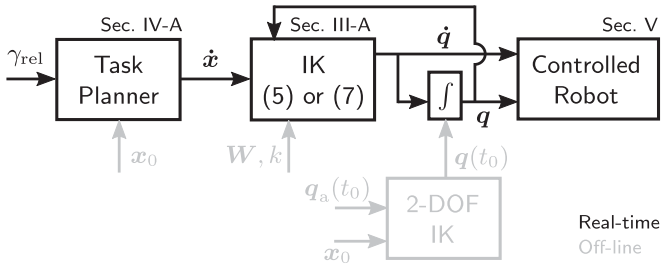

Fig. 2. Example overview: The dynamic optimization problem (10) is solved off-line and returns parameters $\boldsymbol{c}^*$ that keep the system within constraints (1) for a range of $\boldsymbol{x}_0$ and γ_rel.

This automation of finding an appropriate parametrization $\boldsymbol{c}$ off-line for a real-time capable pseudo-inverse kinematic planner yields the following two major benefits: (i) The optimization (10) implicitly penalizes large velocities, which prevents singular Jacobians $\boldsymbol{J}$. (ii) The optimization (10) maximizes the distance to all joint constraints. Therefore, other dynamically challenging tasks $\boldsymbol{x}(t)$ with $t \in [t_0, t_\mathrm{f}]$ can potentially be executed if they are close to the nominal task $\boldsymbol{x}^*(t)$. Simulations in the next section evaluate this potential flexibility.

IV. Illustrative Example

In this section, tracking the trajectory of a planar, fast flying part is chosen as challenging example. The motivation for this example originates in previous works that had to reduce (task) velocity [3], [5] to perform the desired end-effector motion accurately.

Fig. 2 illustrates how the proposed methods connect in the example. A discussion on the real-time flexibility beyond a known nominal trajectory closes the section.

A. Catching Trajectory Generation

Consider the translational task $\boldsymbol{x} \in \mathbb{R}^2$ of tracking a ballistic flight path in the vertical plane. Hence, the first element x_1 and the second element x_2 denote the horizontal and vertical displacement, respectively. Furthermore, the nonprehensile catching task requires the end-effector to accelerate or decelerate at high velocities. Therefore, the following paragraph briefly describes a dynamical system task motion planner that parametrizes the path such that velocities and accelerations only act along this path.

As a start, note that the vertical position x_2 and the heading angle α, cf. Fig. 3, are unique with respect to the horizontal position x_1 given a ballistic part trajectory $\boldsymbol{x}_\mathrm{P} \in \mathbb{R}^2$. As this letter focuses on the joint trajectory planning, not the catching problem, we further assume without loss of generality that part and end-effector have the same initial position $\boldsymbol{x}_\mathrm{P}(t_0) = \boldsymbol{x}(t_0)$ and velocity $\dot{\boldsymbol{x}}_\mathrm{P}(t_0) = \dot{\boldsymbol{x}}(t_0)$. Part dimensions are neglected for the same reason. Moreover, for the purpose of a concise notation, the total end-effector velocity is denoted $v := \|\dot{\boldsymbol{x}}\|_2$ and initial values are abbreviated $v_0 = v(t_0)$ and $\alpha_0 = \alpha(t_0)$. Then, the translational part velocities for given initial values,

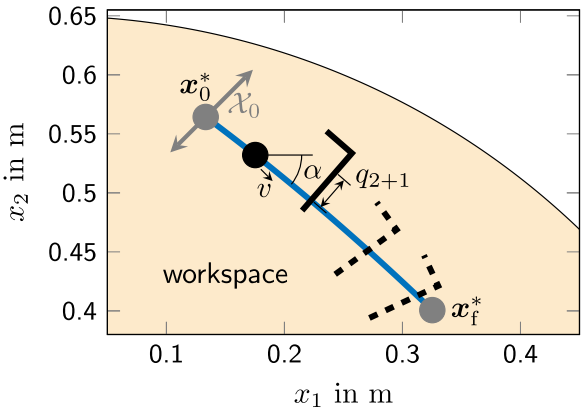

Fig. 3. Segment of the part flight trajectory that must be followed by the end-effector ($\overline{Q}_3 = 1.5\,\mathrm{cm}$) for a controlled catch. The edge of the end-effector [4] compensates for post-impact motion perpendicular to the flight path.

TABLE I
PARAMETER AND CONSTRAINT VALUES

Symbol	Value	Quantity
l_1, l_2	0.32 m	Link lengths
a_1	0.216 m	Center of mass, 1st Link
a_2	0.154 m	Center of mass, 2nd Link
m_1	2.3 kg	Mass of 1st link
m_2	1.3 kg	Mass of 2nd link
I_1	0.0333 kg m^2	Inertia of 1st link
I_2	0.0215 kg m^2	Inertia of 2nd link
g	9.81 ms^{-2}	Gravitational constant
α_0	–39°	Initial heading
$x_1(0)$	0.13 m	Initial horizontal position
$x_2(0)$	0.56 m	Initial vertical position
v_0	4.2 ms^{-1}	Initial velocity
$\overline{\mathbf{T}}_1$ $(\overline{\mathbf{T}})$	(–)50 NM	Peak torque limits
$\overline{\mathbf{V}}_1$ $(\overline{\mathbf{V}})$	(–) 6 rad s^{-1}	Joint velocity limits
$\overline{Q}_3$	{1.5, 5.0} cm	Limit of the augmented joint (end-effector length)

parametrized in terms of the horizontal position, are

$$\dot{x}_{1,\mathrm{P}}(x_1) = v_0 \cos\alpha_0 = \text{const.}, \tag{11}$$

$$\dot{x}_{2,\mathrm{P}}(x_1) = v_0 \sin\alpha_0 - g\frac{x_1 - x_1(t_0)}{\dot{x}_{1,\mathrm{P}}}\,. \tag{12}$$

Hence, the position dependent heading angle becomes

$$\alpha(x_1) = \operatorname{atan}\left(\frac{\dot{x}_{2,\mathrm{P}}(x_1)}{\dot{x}_{1,\mathrm{P}}(x_1)}\right)\,, \tag{13}$$

which takes negative values while the part is in the falling phase. Finally, we define a state vector $\boldsymbol{\xi} := \left[v\ \boldsymbol{x}^T\right]^T \in \mathbb{R}^3$ resulting in the dynamical system motion planner

$$\dot{\boldsymbol{\xi}} = -\begin{bmatrix} -g\sin(\alpha(\xi_2)) \\ \xi_1\cos(\alpha(\xi_2)) \\ \xi_1\sin(\alpha(\xi_2)) \end{bmatrix} + \begin{bmatrix} \gamma_\mathrm{rel} \\ 0 \\ 0 \end{bmatrix}\,,\ \boldsymbol{\xi}_0 = \begin{bmatrix} -v_0 \\ x_1(0) \\ x_2(0) \end{bmatrix}\,, \tag{14}$$

where for now the input $\gamma_\mathrm{rel} = 0$. The purpose of this input γ_rel will become clear in Section IV-C. The dynamical system (14) ensures that an end-effector velocity ξ_1 and its respective acceleration $\dot{\xi}_1$ only act along the part's ballistic path $\boldsymbol{x}_\mathrm{P}$. Solving (14) numerically is computationally inexpensive and thus suits well to potentially process real-time feedback, e.g., from analogue distance sensors integrated in the end-effector.

B. Simulative Method Comparison

Three inverse kinematic methods are compared in simulation:

S1) The non-redundant $(m = 2)$ overarm solution of a 2-DOF manipulator.

S2) The virtually redundant $(m = 3)$ overarm solution of a 2-DOF manipulator using the pseudo-inverse (6) with a diagonal weighting matrix and optimization (10).

S3) The virtually redundant $(m = 3)$ overarm solution of a 2-DOF manipulator using the pseudo-inverse with gradient projection (7), optimization (10), and $w(\boldsymbol{q}) = -\frac{1}{2}(q_3 - \overline{Q}_3/2)^2/(\overline{Q}_3 - \underline{Q}_3)^2$.

The three approaches are evaluated for the same task with two exploitable end-effector lengths $\overline{Q}_{3,a} = 1.5$ cm and $\overline{Q}_{3,b} = 5$ cm, referred to as S1a, S1b, S2a, S2b, S3a, and S3b, respectively. The displacements of the joints q_1 and q_2 are considered unconstrained. The end-effector acceleration along the ballistic path is set to equal the part acceleration $\gamma_\mathrm{rel} = 0$. For the purpose of this letter, we only consider a short segment of $t_f = 60\,\mathrm{ms}$ $(t_0 = 0)$ with a large initial part velocity of $v_0 = 4.2\,\mathrm{ms}^{-1}$ and nominal position $\boldsymbol{x}_0^* = [0.13\,\mathrm{m}\ 0.56\,\mathrm{m}]$ in the manipulator workspace, cf. Fig. 3. Together with $\alpha_0 = -39°$ this corresponds to a zero-height throwing point of $\boldsymbol{x} = [-2.15\,\mathrm{m}\ 0\,\mathrm{m}]^T$. All other parameters and constraints are collected in Table I, whereas velocity and peak torque constraints are considered symmetric.

The left column of Fig. 4 displays velocities and torques, which the given task requires using the well-known 2-DOF overarm inverse kinematics (S1). Horizontal lines, as in all plots that follow, highlight constraints. The length of the virtual joint is added to ℓ_2. Obviously both velocity and torque limits are violated several times, which potentially leads to inaccurate tracking of the desired end-effector trajectory.

The middle column in Fig. 4 shows how the augmented virtual joint is exploited using (5) with the optimization result[1] $\boldsymbol{c}^*_{\mathrm{S2a}} = [0.015\ 31.0\ 21.6\ 177.7]$ and $\boldsymbol{c}^*_{\mathrm{S2b}} = [0.041\ 81.2\ 65.7\ 163.5]$. In comparison to S1, torque requirements are significantly reduced and velocity constraints are met. An increase from 1.5 cm to 5.0 cm in the exploitable end-effector size also shows benefits. For example, the lowest value of the joint velocity (S2a) $\dot{q}_2$ increases from -5.67 rad s^{-1} to -5.14 rad s^{-1}.

The right column in Fig. 4 shows how the augmented virtual joint is exploited using (7) with the optimization result $\boldsymbol{c}^*_{\mathrm{S3a}} = [0.015\ 31.0\ 21.6\ 177.7\ 0]$ and $\boldsymbol{c}^*_{\mathrm{S3b}} = [0.041\ 93.0\ 73.9\ 160.4\ 0.085]$. For the short end-effector $\overline{Q}_{3,a}$ the optimization (10) returns the same result $(k = 0)$ as in the previous case S2a. For the longer end-effector $\overline{Q}_{3,b}$ the optimization returns a small $k = 0.085$. For example, the lowest value of the joint velocity $\dot{q}_1$ in S2b increases from -5.26 rad s^{-1} to -5.22 rad s^{-1}. In both cases, convergence problems of (10) are

[1] using the standard SQP solver of MATLAB

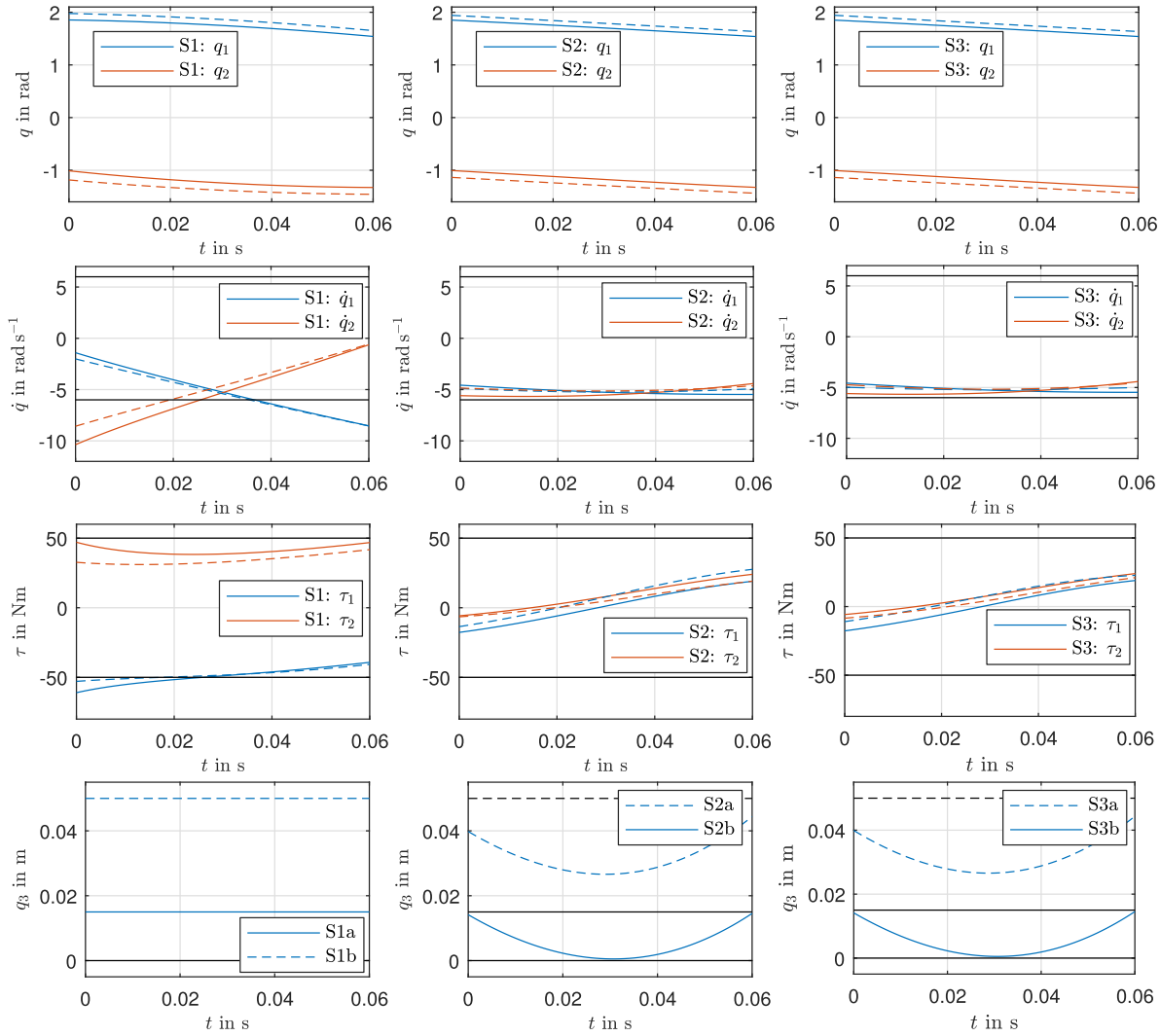

Fig. 4. Simulation results for non-redundant robot (S1), redundancy resolution with weighted pseudo-inverse (S2), and with gradient projection (S3). Solid lines refer to augmentation with the shorter virtual joint $\overline{Q}_{3,a} = 1.5$ cm and the dashed lines to a larger joint $\overline{Q}_{3,a} = 5.0$ cm.

observed. This is mainly due to the textbook-choice of $w(\boldsymbol{q})$, which produces large velocities in the nullspace to force the third joint to its range center. Such behavior is not necessarily advantageous, see also [15].

Because of the above observations, we limit the following discussion to S2a and S2b.

C. *Flexibility of Solutions and Discussion*

Referring to the introduction, the goal of this letter is to provide joint motion planning on, or close to a given path to enable real-time acceleration input. Hence, we raise the question: Given an optimization-based pseudo-inverse solution S2a or S2b, to what extent does such a solution apply for varying accelerations? For this purpose, we introduced $\gamma_{\mathrm{rel}} \in \mathbb{R}$ as time-varying input to (14). As an example, choosing $\gamma_{\mathrm{rel}} = g$ would cause a relative acceleration of g between part and end-effector.

Fig. 5 illustrates the effect of varying $\gamma_{\mathrm{rel}} \in [-g, 6g]$ in steps of g at the example of q_3 and τ_1 for S2a. Using the nominal solution of $\boldsymbol{c}^*$ as above for $\gamma_{\mathrm{rel}} = 0$, joint trajectories stay within constraints for $\gamma_{\mathrm{rel,S2a}} \in [-0.1g, 4.9g]$ and γ_{rel}, S2b $\in [-1.1g, 5g]$. Accelerations $\gamma_{\mathrm{rel}} > 5g$ violate the torque limit $\overline{T}_1$ and accelerations $\gamma_{\mathrm{rel}} < 0.1g$ (or $-1.1g$) violate the constraint $\overline{Q}_3$. Further simulations show that all positive initial velocities $v_0 < 4.2\,\mathrm{ms}^{-1}$ also meet the constraints.

Manipulation systems in less structured environments may also require executing a primitive starting at several points in the manipulator workspace, e.g., catching of human throws [5]. Hence, we here ask if an area around $\boldsymbol{x}_0^*$ exists for which the solution $\boldsymbol{c}^*$ stays within all constraints. For this purpose, we perform the simulation for various initial positions on a line (see Fig. 3): $\boldsymbol{x}_0 \in \mathcal{X}_0(x_0^*, \mathcal{I}) = \{\boldsymbol{x} | x_2 = x_2^*(0) + (d - x_1^*(0)), d \in \mathcal{I} \subset \mathbb{R}\}$. The flexibility with respect to $\boldsymbol{x}_0$ (length of the set $\mathcal{I}$) highly depends on the degree of exploitation of q_3, whereas the influence on velocity and torque requirements is almost negligible. For $\overline{Q}_{3,a}$ the set of feasible d around $x_1^*(0)$ is much smaller than 1 cm. For $\overline{Q}_{3,b}$ the feasible set becomes $\mathcal{I} = \{d | 0.12 \leq d \leq 0.21\}$ cm. Hence, one

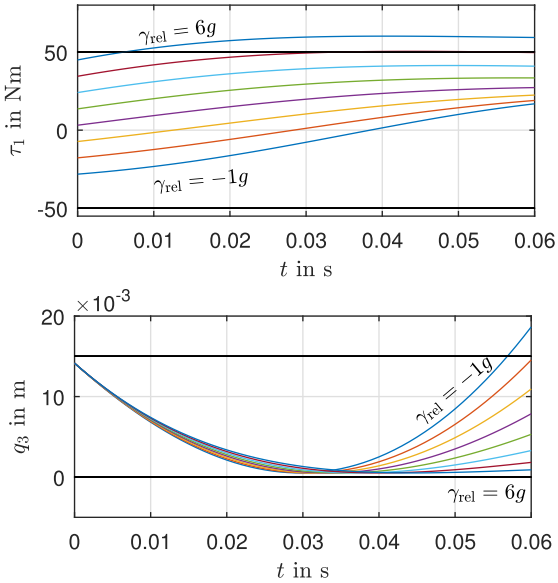

Fig. 5. The effect on required joint torque and joint exploitation when applying various relative accelerations $\gamma_{\mathrm{rel}} \in \{-g, 0g, g, \ldots, 6g\}$ between part and end-effector.

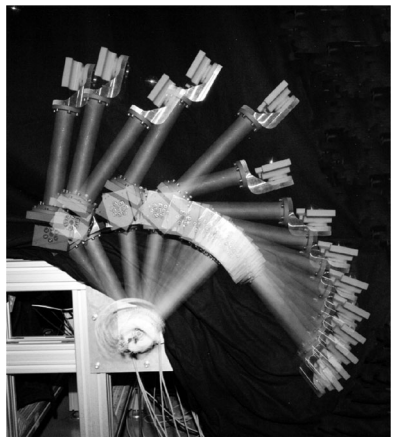

Fig. 6. Stroboscopic visualization of a nonprehensile catching motion with a non-redundant robot.

Fig. 7. Exploitation of the virtual nonprehensile joint q_{2+1} (c.f. Fig. 3) of length $\overline{Q}_{3,a}$ during flight trajectory tracking illustrated by three frames taken at approximately 0 ms, 30 ms and 60 ms. Ball and end-effector have approximately the same initial velocity of $v_0 = 4.2\ \mathrm{ms}^{-1}$.

could partition a space of potential initial states and calculate an optimal parametrization for each part off-line. Or, as in this particular case of planar part catching, one-dimensional manifolds for several sets $\mathcal{X}_0$. Alternatively, another off-line optimization run could exploit the larger end-effector to follow the fast flight trajectory longer (larger t_{f}) up to 85 ms instead of 60 ms.

Besides the re-computation of $\boldsymbol{c}$ in less structured environments, the proposed approach is also limited by the numerical integration depicted in Fig. 2. In a real-time scenario, such integration uses fixed step-size solvers that cannot provide the reliable accuracy of variable step-size solvers. The same problem applies to our task planner that has to perform similar integrations when solving (14) in real-time. For the above described scenario with $t_{\mathrm{f}} = 60$ ms the Euclidean workspace error remains in the negligible order of 10^{-6} m using the Runge-Kutta method with a fixed step-size of 1ms. If numerical errors reach non-negligible magnitudes, closed-loop differential inverse kinematic planners (e.g., [22]) should be used instead.

Hence, the simple method (5) with off-line optimized weights is already well suited for kinematic trajectory planning in the presence of dynamically unconstrained nonprehensile joints. Even with short, virtual prismatic joints, trajectories that are highly unfeasible in the classical sense may now be executed with relative accelerations on the ballistic path that change in real-time. Therefore, the optimization based pseudo-inverse method in this letter potentially enables fast and truly soft [5] ballistic catching with low impact velocities, if an appropriate task level controller for input γ_{rel} is found.

V. Experimental Validation

Here we verify the dynamic feasibility claimed with the above simulations using a 2R planar robot vertically mounted as shown in Fig. 6 and with parameters given in Table I. A simple high-gain PD-controller operates the robot at 1 kHz. If we let the joint displacement error and joint velocity error be $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$, respectively, the control law is $\boldsymbol{\tau} = K_{\mathrm{P}}\boldsymbol{e} + K_{\mathrm{D}}\dot{\boldsymbol{e}}$ with $K_{\mathrm{P}} = 12000$ and $K_{\mathrm{D}} = 100$. The joints consist of RE40 Maxon DC motors, MR Maxon (type L) 1024-bit encoders, and HFUC Harmonic Drive 1:100 gears. Hence, the joint displacement on the load side is measured with an accuracy of $1.5 \cdot 10^{-5}$ rad. Off-line solving of two point boundary value problems serves as joint motion planner to reliably move the robot from rest to the start of the tracking phase at t_0 and from the end t_f to a resting position. The ball in Figs. 6 and 7 is thrown in a repeatable way by another robot as described in [4], which relates to the assumption of a known path.

For the fast trajectory tracking experiment, we implemented the motion planner as in Fig. 2, where the initial states and $\boldsymbol{c}^*_{\mathrm{S2a}}$ are identical to the above simulation. The input γ_{rel} in the first half $t \in [0, 30)$ ms constantly equals zero, which was also assumed in the off-line optimization that resulted in $\boldsymbol{c}^*_{\mathrm{S2a}}$. In the second half, the ramp $\gamma_{\mathrm{rel}}(t) = 4g\frac{t-30\,\mathrm{ms}}{30\,\mathrm{ms}}$ for $t \in [30, 60]$ ms simulates a varying input that requires the robot to decelerate such that the relative acceleration between object and end-effector increases to $\gamma_{\mathrm{rel}}(t_f) = 4g$. Such variation was not considered during the off-line optimization (10) but lies well within the bounds discussed in the last section.

Fig. 7 and the multimedia attachment illustrate the resulting exploitation of the virtual nonprehensile joint q_{m+1}. Note here that only the end-effector position ($n = 2$) and not its orientation is part of the task planner. Preliminary experiments in [4] showed that this can be sufficient for nonprehensile catching.

Based on ten trials, Fig. 8 depicts the errors of the two actuated joints q_1 and q_2 during the tracking phase. The very low values

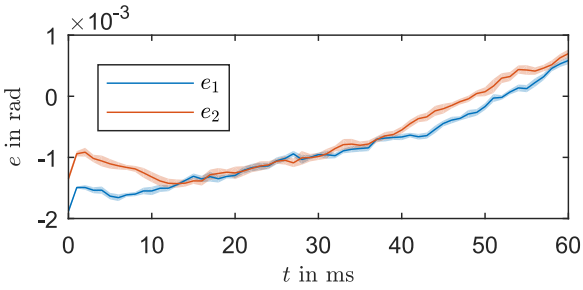

Fig. 8. Mean joint errors of ten trials of flight trajectory tracking, where the shaded areas indicate twice the standard deviation. From these errors results a worst case Euclidean error in the workspace of less than 2 mm.

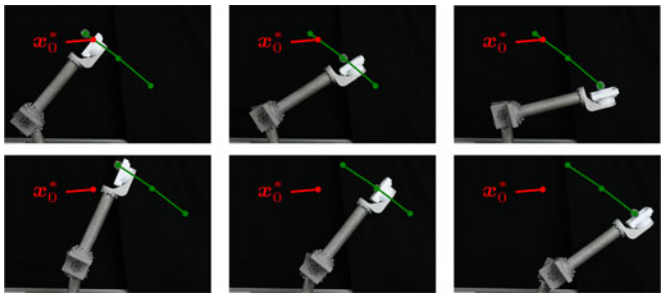

Fig. 9. Tracking of fast ballistic trajectories with a long ($\overline{Q}_{3,b}$) nonprehensile end-effector, acceleration $\gamma_{\mathrm{rel}} = 0$ ms^{-2} for $t \in [0, 60]$ ms, and $v_0 = 4.2$ ms^{-1}. Frames are taken at approximately 0 ms, 30 ms and 60 ms. The upper row shows the reference motion ($d = 0.13$ cm), whereas the lower row shows accurate tracking from a different starting point ($d = 0.21$ cm) while using the same $\boldsymbol{c}^*_{\mathrm{S2b}}$, see also Fig. 3.

of these joint errors relate to workspace errors below 2 mm and thus confirm dynamically feasible trajectory planning by (5) and (10). Especially from $t = 30$ ms, where we switched from known to unforeseen acceleration, no negative effects on the tracking performance occur.

The flexibility with respect to the initial state illustrates Fig. 9. Here, a nonprehensile end-effector of length $\overline{Q}_{3,b}$ allows to track fast trajectories from different initial points without recomputation of $\boldsymbol{c}^*_{\mathrm{S2b}}$. As a result, future work in nonprehensile manipulation with intermittent contacts may now be able to maintain stability or robustness claims, made on task level, also in experiments.

VI. Conclusion and Future Work

Dynamically unconstrained joints have been introduced for intermittent nonprehensile dynamic manipulation to enable fast and accurate execution of primitives like catching, batting or juggling. A weighted pseudo-inverse approach with optimized weights and initialization was shown to be effective for this purpose. The experimental evaluation showed a non-redundant robot that could accurately follow the fast flight trajectory of a part for 60 ms by exploiting a short unconstrained joint of only 1.5 cm length. The method introduced in this letter potentially enables catching robots, for the first time, to control the catch in the last moments before collision. In our future work, we plan to use the new input by building analog short-range sensors into the end-effector. One would thus close the feedback loop on task level subject to the dynamic feasibility constraints.

Acknowledgment

The authors would like to thank T. Weber for her preliminary studies, which motivated this work.

References

[1] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2592–2599.
[2] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1049–1065, Oct. 2014.
[3] G. Bätz, A. Yaqub, H. Wu, K. Kühnlenz, D. Wollherr, and M. Buss, "Dynamic manipulation: Nonprehensile ball catching," in *Proc. 18th Mediterranean Conf. Control Autom.*, 2010, pp. 365–370.
[4] M. M. Schill, F. Gruber, and M. Buss, "Quasi-direct nonprehensile catching with uncertain object states," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2468–2474.
[5] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 1–10, Apr. 2016.
[6] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *Int. J. Robot. Res.*, vol. 18, no. 1, pp. 64–92, 1999.
[7] K. M. Lynch and T. D. Murphey, "Control of nonprehensile manipulation," in *Control Problems in Robotics (Springer Tracts in Advanced Robotics)*. New York, NY, USA: Springer-Verlag, 2003, vol. 4, pp. 39–57.
[8] J.-C. Ryu, F. Ruggiero, and K. M. Lynch, "Control of nonprehensile rolling manipulation: Balancing a disk on a disk," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1152–1161, Oct. 2013.
[9] V. Lippiello, F. Ruggiero, and B. Siciliano, "The effect of shapes in input-state linearization for stabilization of nonprehensile planar rolling dynamic manipulation," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 492–499, Jan. 2016.
[10] S. Schaal and C. G. Atkeson, "Open loop stable control strategies for robot juggling," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1993, pp. 913–918.
[11] K. M. Lynch and C. K. Black, "Recurrence, controllability, and stabilization of juggling," *IEEE Trans. Robot. Autom.*, vol. 17, no. 2, pp. 113–124, Apr. 2001.
[12] P. Reist and R. D'Andrea, "Design and analysis of a blind juggling robot," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1228–1243, Dec. 2012.
[13] R. L. Andersson, "Aggressive trajectory generator for a robot ping-pong player," *IEEE Control Syst. Mag.*, vol. 9, no. 2, pp. 15–21, Feb. 1989.
[14] M. Gardner, Y.-B. Jia, and H. Lin, "Batting flying objects to the target in 2D," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3225–3232.
[15] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 286–292, Apr. 1995.
[16] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern.*, vol. 7, no. 12, pp. 868–871, Dec. 1977.
[17] F. Flacco, A. de Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 637–654, Jun. 2015.
[18] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," *J. Dyn. Syst., Meas. Control*, vol. 106, no. 1, pp. 102–106, 1984.
[19] G. Antonelli, S. Chiaverini, and G. Fusco, "Kinematic control of redundant manipulators with on-line end-effector path tracking capability under velocity and acceleration constraints," *IFAC Proc. Vol.*, vol. 33, no. 27, pp. 183–188, 2000.
[20] A. Pekarovskiy, T. Nierhoff, S. Hirche, and M. Buss, "Dynamically consistent online adaptation of fast motions for robotic manipulators," *IEEE Trans. Robot.*, doi: 10.1109/TRO.2017.2765666.
[21] H. Zghal, R. V. Dubey, and J. A. Euler, "Efficient gradient projection optimization for manipulators with multiple degrees of redundancy," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1990, pp. 1006–1011.
[22] B. Siciliano, *Robotics: Modelling, Planning and Control (Advanced textbooks in control and signal processing)*. London, U.K.: Springer, 2009.