A Dynamic Programming Framework for Optimal Planning of Redundant Robots Along Prescribed Paths With Kineto-Dynamic Constraints

Enrico Ferrentino◦, *Member, IEEE*, Heitor J. Savino, *Member, IEEE*, Antonio Franchi◦, *Fellow, IEEE*, and Pasquale Chiacchio◦, *Senior Member, IEEE*

Abstract— Offline optimal planning of trajectories for redundant robots along prescribed task space paths is usually broken down into two consecutive processes: first, the task space path is inverted to obtain a joint space path, then, the latter is parametrized with a time law. If the two processes are separated, they cannot optimize the same objective function, ultimately providing sub-optimal results. In this paper, a unified approach is presented where dynamic programming is the underlying optimization technique. Its flexibility allows accommodating arbitrary constraints and objective functions, thus providing a generic framework for optimal planning of real systems. To demonstrate its applicability to a real world scenario, the framework is instantiated for time-optimality on Franka Emika's Panda robot. The well-known issues associated with the execution of non-smooth trajectories on a real controller are partially addressed at planning level, through the enforcement of constraints, and partially through post-processing of the optimal solution. The experiments show that the proposed framework is able to effectively exploit kinematic redundancy to optimize the performance index defined at planning level and generate feasible trajectories that can be executed on real hardware with satisfactory results.

Note to Practitioners—The common planning algorithms which consolidated over the years for generating trajectories for non-redundant robots are not adequate to fully exploit the more advanced capabilities offered by redundant robots. This is especially true in performance-demanding tasks, as for robots employed on assembly lines in manufacturing industries, repeatedly performing the same activity. Once the assembly line engineer has defined the tool path in the task space, our planning algorithm unifies inverse kinematics and time parametrization so as to bring the manipulator at its physical limits to achieve specific efficiency goals, being execution time the most typical one. The algorithm is configurable in terms of constraints to consider and objective functions to optimize, therefore it can be easily adapted to optimize other custom-defined efficiency indices, to better respond to the needs of the automation plant. Being based on discrete dynamic programming, the global optimum is guaranteed for a given resolution of the problem. This can be configured by the operator to achieve the desired trade-off between efficiency and planning time. In our experiments, we go through the whole process of planning and executing a time-optimal trajectory on a real robot, and discuss some practical details, such as trajectory smoothness and actuator saturation, aiding the practitioners in deploying our algorithm effectively. Currently, the algorithm's applicability is limited to those cases where hours are available for planning, hence it is not well-suited for those cases where the robot activity has to change frequently. By replacing the underlying dynamic programming engine with a different methodology, such as randomized algorithms, the planning time could be controlled to be upper-bounded, thus returning the most efficient solution that can be achieved in the time available for reconfiguring the production. Other applications of interest include optimal ground control of space robotic assets and performance benchmarking of online planning algorithms.

Index Terms—Robot programming, manipulator motion-planning, time optimal control, optimization methods.

I. INTRODUCTION

A. Scope and Objective

KINEMATICALLY redundant manipulators possess more degrees of freedom than those strictly required to execute a given task. Such a characteristic gives the system a higher degree of dexterity and mobility that can be exploited to optimize performance indices of interest, besides fulfilling the main task. Redundancy can be structural, when the robot is designed to be redundant for generic tasks, or functional, when the robot is operated so as to be redundant. The concept applies to anthropomorphic serial chains, that are by far the most widespread robotic systems in manufacturing industries, but it is general enough to be extended to more complex systems, including humanoids, mobile manipulators (terrestrial, aerial), parallel robots and systems of cooperating robots. The optimization problem that looks into the exploitation of redundancy to optimize a given performance index is commonly referred to as *redundancy resolution*.

Many robotic tasks require to follow a specific path, such as in welding, cutting, gluing and in some assembly and
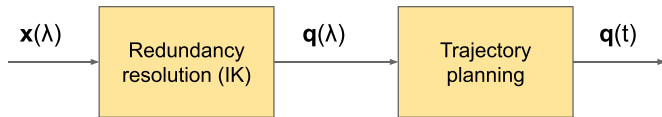
Fig. 1. Redundancy resolution and trajectory planning as two independent processes; $\mathbf{x}(\lambda)$ is the task space path, $\mathbf{q}(\lambda)$ the joint space path and $\mathbf{q}(t)$ the joint space trajectory.
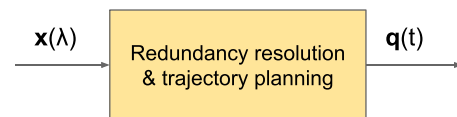

Fig. 2. Redundancy resolution and trajectory planning as one unique process; $\mathbf{x}(\lambda)$ is the task space path and $\mathbf{q}(t)$ the joint space trajectory.

disassembly tasks. Planning a collision-free task space path is also a convenient solution when the environment is cluttered. This is opposed to *point-to-point* (PTP) motion, where the robot links are more loosely constrained during motion. The definition of the speed by which the path is tracked is also a decision variable that, in most applications, is necessary to optimize. The typical objective is time minimization, but any performance index involving derivatives can be designed for a specific purpose. The optimization problem that looks into the definition of velocity along a given path is commonly referred to as *optimal trajectory planning* or *optimal time-parametrization*.

Both the optimization problems above can be addressed (and solved) at planning level, sometimes offline, before the robot moves, or at control level, while the task is being executed and the system has to react to unforeseen events. Although, nowadays, most applications require the robots to live and operate in unstructured, unknown and highly-dynamical environments, there still are many situations in which tasks are better planned offline. One of the most frequent employments of anthropomorphic arms in manufacturing industries concerns the execution of repetitive tasks in structured environments, where the task is planned once and executed cyclically. In aerospace, offline planning instances exist in the mission design phase for feasibility and budget assessments. Also, some sequences for in-orbit manipulation can be pre-planned in an optimal way. In missions that are characterized by windowed communications, a long planning time is usually available on ground to deliver more efficient commands to the spaceborne asset. On the contrary, when the domain requires highly-reactive behaviors, offline planning can provide benchmark solutions to evaluate the performance of online control algorithms. The scenarios just described are the domain of this paper.

Whether they are executed on-line or offline, redundancy resolution and trajectory planning have been mostly treated as two independent, consecutive optimization processes [1], [2], [3], as depicted in Fig. 1. A task space path is given, that is mapped, through inverse kinematics, in the joint space (possibly optimizing a geometric quality index), then, the joint space path is time-parametrized to yield a trajectory (respecting constraints and possibly optimizing a different quality index), that can be executed through motion control at joint level. The optimization performed at the former stage can be functional to the optimization performed at the latter. For instance, if the objective of trajectory planning is to define a minimum-time motion, redundancy resolution may optimize, according to a heuristic approach, the acceleration/deceleration capabilities of the manipulator along the path. Overall, the two processes "cooperate" to achieve a common objective.

Unfortunately, the resolution of two independent optimal problems does not guarantee the achievement of the overall global optimum. Indeed, if the objective is to optimize a generic dynamic index, e.g., traveling time, but the time law is only defined at the level of trajectory planning, there is no possibility to formulate a redundancy resolution problem which explicitly considers it.

Hence, the objective of this paper is to provide a unified solution for both optimization problems, as in Fig. 2, so that kinematic redundancy is effectively exploited for the minimization/maximization of the objective function defined at trajectory planning level.

B. Related Works

Optimal trajectory planning along prescribed paths deals with the optimization of an integral cost function, subject to the kinematic constraint of the end-effector's path and dynamic capabilities of the actuators (mainly torque and torque rate limits). The foundations of the problem were laid for time minimization and non-redundant robots [4], [5], [6], [7], where the unique joint space path is parametrized with respect to the curvilinear coordinate of the path λ. The robot kinematic and dynamic constraints are encoded on limits on its first- and second-order time-derivatives $\dot{\lambda}$ and $\ddot{\lambda}$, called *pseudo-velocity* and *pseudo-acceleration*, respectively. In this new domain, λ and $\dot{\lambda}$ represent the state variables and $\ddot{\lambda}$ the input of a parametrized dynamic system. The optimal solution is bang-bang in $\ddot{\lambda}$ [8]. If, for a redundant manipulator, the joint space path is already assigned, the theory of non-redundant manipulators holds without modifications.

The first extension to kinematically redundant systems, that includes the more generic case of a path assigned in the task space, was developed in [9] and [10]. Therein, a non-minimal extended state is considered, made of λ, $\dot{\lambda}$, the joint positions and their derivatives. The input corresponds to actuation torques. The time-optimal problem is solved numerically by making use of the *Extended Pontryagin's Maximum Principle (EPMP)* and considering an approximation of the torque constraints. This formulation is inflexible with respect to the introduction of additional constraints and objective functions.

By adopting a suitable parametrization of redundancy [11], e.g. through *joint space decomposition (JSD)* [12], a minimal state of the parametrized dynamic system can be found, composed of λ, $\dot{\lambda}$, the redundancy parameter, and its first-order derivative. This time, the equations can be arranged to let the input be the vector made of $\ddot{\lambda}$ and the acceleration of the redundancy parameter, but they cannot be constrained explicitly. As a consequence, a linear programming (LP) problem has to be defined for each possible state in order to determine the optimal input, that is extremized according to the bang-bang paradigm. In [13], in order to provide an initial state, the initial joint positions are fixed and the resolution is achieved through a try-and-error approach. In addition, in order to make the algorithm more efficient, the number of switching

points is possibly reduced with respect to the real time-optimal trajectory. Objective functions other than tracking time cannot be considered, as the bang-bang hypothesis is made. Lastly, the algorithm cannot enforce zero joint velocity at the end of the motion, which is undesirable in most applications.

A different approach consists in addressing the optimization problem directly [14], [15], [16], [17], without any prior manipulation of the quantities in play. The time axis is divided into a fixed number of intervals of the same variable length, whose sum is t_f, that is the result of the optimization process. When t_f is given, the duration of each time interval in the time domain is also determined. The problem is to find, for each of the intervals, the value of λ, the joint position vector, and their derivatives. The path constraint is imposed through second-order kinematics, either with JSD or null-space projection. For both techniques, since the path constraint is enforced through numerical integration of the second-order solution, an additional positional kinematics constraint needs to be included to prevent drift. The problem can be solved with direct multiple shooting using the interior-point method, after an initial guess on the optimizing variables is made. Since the problem is non-convex, globally-optimal solutions cannot be guaranteed and, in general, the quality of the resulting solution strictly depends on the initial guess. On the other hand, the formulation is flexible enough to include constraints on the joint positions and their derivatives, as well as other application-specific constraints. For instance, in [17], the path constraint is further derived to account for the limits on joint positions, velocities, accelerations, jerks and snaps (i.e. the fourth-order derivative of the position) through which continuous and quasi-differentiable motor torques can be obtained. By acting directly on $\dddot{\lambda}$, this is also the objective in [18]. This result is of utmost importance since the insufficient continuity of globally-optimal minimum time trajectories is the main shortcoming when it comes to the control of real robots.

For the sake of completeness, we note that a multitude of different optimization approaches exist in more recent literature on trajectory planning. They include dynamic optimization in the form of convex quadratic [19], [20] or cone [21] programming, evolutionary algorithms [22], [23], [24], [25], dynamic programming [26], heuristic algorithms [27], and machine learning [28], with several approaches to describe the solution, in the form of, e.g., polynomials [27], [29] and B-splines [22], [23], [24], [25] to control the joint space smoothness or achieve multi-objective optimization [20], [22], [23], [24], [25], [30]. The mentioned techniques, however, are only applied to solve sub-problems of the unified optimal redundancy resolution and time parametrization problem addressed here. They consider, for instance, PTP instead of path-constrained tasks [23], [25], or only the first [19], [31] or second [21], [29], [30] stage in Fig. 1.

C. Methodology and Contribution

In all the offline planning applications that motivate this work, although the environment where the robot operates is structured, static, known or partially known, not all the formulations of the optimization problem can capture the complexity of the real system and some show greater flexibility than others. This paper adopts discrete dynamic programming

(DDP) as the main underlying methodology and central idea to cope with the complexity of reality. Also, because of its employment, in previous works, for both the optimization problems described above, it naturally arises as the unifying approach (see, for instance, [32], [33], [34], [35], [36] for redundancy resolution, and [37], [38], [39], [40] for trajectory planning). DDP is extremely flexible in the accommodation of arbitrary constraints and objective functions and, compatibly with the available planning time, can achieve the global optimum. When applied to real systems, its formulation can well describe the constraints at hand and no conservative hypotheses are necessary. The computational complexity of DDP is square with the cardinality of the discrete sets defining the feasible search space [36]. However, the more constraints are defined in the problem formulation (which is the typical case of real systems), the more states are excluded from the search. Therefore, differently from several other numerical optimization methods, e.g., those based on the Newton's method, where more constraints make the search of the optimum cumbersome, in DDP they are beneficial to reducing the computation time.

DDP is based on the discretization of the state space which, in principle, complicates the generation of smooth references for motion control. However, the possibility to include higher-order constraints mitigates this issue.

Our contribution is a unified generic framework for redundancy resolution and trajectory planning based on DDP. With respect to previous works proposing a unified solution, i.e. [12], [14], [15], [16], and [17], our methodology does not need the user to provide a seed state, i.e., an initial state from which the search starts, and which closely influences the final solution, but can retrieve the globally-optimal solution of the discretized state space. Possibly, this solution can be used as a valuable seed state for a different optimization problem to further improve other features of the solution, such as its smoothness. We further elaborate on this aspect in Section IV-E.

D. Overview

In Section II, the problem is presented and formalized. First, parameters and equations that describe the problem are introduced in Section II-A and discretized so as to identify a suitable state that provides a minimal description. Then, in Section II-B, the constraints are introduced as maps that enable transitions towards certain states, in a generic framework. Last, in Section II-C, the optimization problem is introduced in a generic way and specialized for time-optimal planning. The algorithmic formulation is provided in Section III, together with some important considerations of practical interest. The results of the experiments are presented in Section IV. First, the experimental setup is described in Section IV-A, then, results of offline planning and execution on a real robot are given in Section IV-B and Section IV-F respectively. In Sections IV-C and IV-D, our approach is compared with two algorithms from [1], [14], [15], [16], and [17], using the decoupled approach of Fig. 1 and another unified approach as in Fig. 2, respectively. The work is concluded in Section V, where some lines of development are also outlined for future research.

II. Optimal Planning of Redundant Robots With Dynamic Programming

A. State Definition

Let us consider a geometrical path for the end-effector $\mathbf{x}(\lambda)$: $[0, L] \to \mathbb{R}^m$, such that a given manipulator with n degrees of freedom is redundant, i.e. $n > m$. It follows that $r = n - m$, the *redundancy degree*, is strictly greater than zero. λ is the arc length, L is the path length. The joint positions along the path $\mathbf{q}(\lambda) \in \mathbb{R}^n$ are unknown, as well as the time law $\lambda(t) : \mathbb{R} \to [0, L]$ associated with them. The time derivative of the arc length $\dot{\lambda} = \frac{d\lambda}{dt}$ encodes the time information, and, when expressed as $\dot{\lambda}(\lambda)$, for variable λ, it is often referred to as *phase plane trajectory* (PPT) [4], [5]. Intuitively, the PPT corresponds to the norm of the tangential velocity with respect to the arc length.

Let us discretize λ in space with a given step Δ_λ and $N_i \in \mathbb{N}$, i.e.,

$$\lambda(i) = i\Delta_\lambda, \quad \text{with } i = 0, 1, 2, \ldots, N_i \quad \text{and} \quad \Delta_\lambda = \frac{L}{N_i} \quad (1)$$

such that $\mathbf{x}(\lambda(i)) = \mathbf{x}(i)$ can be viewed as the $(i + 1)$-th waypoint. The variable i is termed *stage index*.

The robot dynamic model is given by

$$\boldsymbol{\tau}(i) = \mathbf{H}(\mathbf{q}(i))\ddot{\mathbf{q}}(i) + \mathbf{f}(\mathbf{q}(i), \dot{\mathbf{q}}(i)), \quad (2)$$

where, by omitting the dependence on i for simplicity, $\mathbf{H}(\mathbf{q})$ is the $n \times n$ inertia matrix and

$$\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \dot{\mathbf{q}}^T \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{d}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}). \quad (3)$$

$\mathbf{C}(\mathbf{q})$ is the $n \times n \times n$ matrix of Christoffel symbols, $\mathbf{d}(\dot{\mathbf{q}})$ is the $n \times 1$ vector of Coulomb and viscous friction terms, and $\mathbf{g}(\mathbf{q})$ is the $n \times 1$ vector of gravitational torques.

Let $S_{(\cdot)}(i)$ be the discrete-time function (or *sequence* of samples) of the continuous-time function $(\cdot)(\lambda)$ up until the i-th sample. For instance, $S_{\dot{\lambda}}(i) = \{\dot{\lambda}(0), \ldots, \dot{\lambda}(i)\}$ is the discrete-time phase plane trajectory up until stage i.

The joint velocities and accelerations at a given stage i, $\dot{\mathbf{q}}(i)$ and $\ddot{\mathbf{q}}(i)$, respectively, in a discrete-time formulation, can be approximated, through fixed-step integration schemes, as functions of the sequences $S_{\mathbf{q}}(i)$, $S_{\dot{\mathbf{q}}}(i)$ and $S_t(i)$, i.e.,

$$\dot{\mathbf{q}}(i) \sim f_{\dot{\mathbf{q}}}(S_{\mathbf{q}}(i), S_t(i)), \quad (4)$$

$$\ddot{\mathbf{q}}(i) \sim f_{\ddot{\mathbf{q}}}(S_{\dot{\mathbf{q}}}(i), S_t(i)). \quad (5)$$

The exact form of the functions above depends on the specific integration scheme. We note here, and further clarify in Section III, that derivatives are assumed to be computed with past samples, in a backward integration scheme, because future samples, in a forward optimization scheme, are unknown, as they depend on optimal choices that yet have to be made.

In the same way, each timestamp $t(i)$ in $S_t(i)$ can be derived from the phase plane trajectory up until stage i, through some discrete approximation, e.g. the trapezoidal approximation used in [38], that generically is

$$t(i) \sim f_t(S_\lambda(i), S_{\dot{\lambda}}(i)). \quad (6)$$

Let us remark, once again, that $S_{\dot{\lambda}}(i)$ is unknown and, as a consequence, $t(i)$ is unknown, too.

Let us define the *forward kinematics function* $\mathbf{k}(\mathbf{q})$: $\mathbb{R}^n \to \mathbb{R}^m$. The path constraint at the given waypoints is given by

$$\mathbf{x}(i) = \mathbf{k}(\mathbf{q}(i)). \quad (7)$$

In order to parametrize the redundancy, we may adopt a joint selection (or JSD) or a joint combination scheme [11], i.e. the redundancy parameters are selected as a subset of the joint positions or they are a more generic function of them. Under this hypothesis, each of the joint positions in $S_{\mathbf{q}}(i)$ in (4) can be computed from non-redundant inverse kinematics (IK) as

$$\mathbf{q}(i) = \mathbf{k}^{-1}(\mathbf{x}(i), \mathbf{v}(i), g(i)), \quad (8)$$

where $\mathbf{v} \in \mathbb{R}^r$ is the vector of redundancy parameters. Since more than one configuration may satisfy the generic non-redundant IK, g is a selector of the specific IK solution to be considered at stage i. For example, if the square inverse kinematics function $\mathbf{k}^{-1}(\mathbf{x}, \mathbf{v})$ admits 4 solutions, $g \in \{0, 1, 2, 3\}$. This way defined, $\mathbf{q}(i)$ is unique. Since $\mathbf{x}$ is a function of λ, we may rewrite the equation above as

$$\mathbf{q}(i) = \mathbf{k}^{-1}(\lambda(i), \mathbf{v}(i), g(i)). \quad (9)$$

Now, we may fold (6) and (9), for each i, into (4) and (5), yielding

$$\dot{\mathbf{q}}(i) \sim f_{\dot{\mathbf{q}}}(S_\lambda(i), S_{\dot{\lambda}}(i), S_{\mathbf{v}}(i), S_g(i)), \quad (10)$$

$$\ddot{\mathbf{q}}(i) \sim f_{\ddot{\mathbf{q}}}(S_\lambda(i), S_{\dot{\lambda}}(i), S_{\mathbf{v}}(i), S_g(i)). \quad (11)$$

By substituting both equations above into (2), we find that, in general, the input is a function of the four identified sequences, i.e.

$$\boldsymbol{\tau}(i) \sim f_{\boldsymbol{\tau}}(S_\lambda(i), S_{\dot{\lambda}}(i), S_{\mathbf{v}}(i), S_g(i)). \quad (12)$$

This means that, given the current state at stage i, selecting the input $\boldsymbol{\tau}(i)$ that drives the discrete-time system to another state is equivalent to selecting the parameters $\dot{\lambda}$, $\mathbf{v}$ and g at the next stage. This is coherent with the logic of discretizing the state space instead of the input space that is common to all the previous works using DDP for redundancy resolution and time-optimal planning. Notoriously, this is also a way to control the number of involved variables due to the addition of extra dimensions, i.e. the *curse of dimensionality* [41], that is a well-known issue in dynamic programming. The parameters $\dot{\lambda}$, $\mathbf{v}$ and g also constitute a minimum representation of the state for each stage, as opposed to [9] and [14] that, rather, use redundant representations. In a formulation based on calculus of variations, this result is related to the minimum number of differential equations required to fully represent the system at hand [11].

Thus, let us proceed by discretizing the variables $\dot{\lambda}$ and $\mathbf{v}$. The former is

$$\dot{\lambda}_l = l\Delta_{\dot{\lambda}}, \quad \text{with } l = 0, 1, 2, \ldots, N_l \quad \text{and} \quad \Delta_{\dot{\lambda}} = \frac{\dot{\lambda}_M}{N_l}, \quad (13)$$

where $\dot{\lambda}_M$ is the maximum pseudo-velocity value that the phase plane trajectory can reach. The latter is

$$\mathbf{v} = \mathbf{j} \circ \Delta_v + \mathbf{v}_{min}, \quad (14)$$

where '$\circ$' denotes the element-wise Hadamard product, $\Delta_v = [\Delta_{v,1}, \ldots, \Delta_{v,r}]$ is the vector of the sampling intervals,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

FERRENTINO et al.: DYNAMIC PROGRAMMING FRAMEWORK FOR OPTIMAL PLANNING OF REDUNDANT ROBOTS 5

$\mathbf{j} \in \mathbb{N}^r$ the vector of the redundancy parameter indices, and $\mathbf{v}_{min}$ is the vector of lower bounds of the redundancy parameters domains. The elements of $\mathbf{j}$ take the maximum values $N_{j,1}, \ldots, N_{j,r}$ so that $[N_{j,1}, \ldots, N_{j,r}] \circ \Delta_v + \mathbf{v}_{min}$ equals the upper bound of the redundant joint domains.

The discretized domains of λ, $\dot{\lambda}$, $\mathbf{v}$ and g (which is naturally discrete) makes up a grid of $r + 3$ dimensions. We essentially adopt the same approach as other DDP formulations for redundancy resolution and time-optimal planning, but we increase the number of dimensions to take into account: (1) the resolution of both problems together; (2) the presence of multiple inverse kinematics solutions.

Let us recall that, for non-redundant manipulators, the curvilinear coordinate parametrization of dynamics is obtained through the following equations:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{x}'\dot{\lambda},$$
$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}\big[\mathbf{x}'\ddot{\lambda} + \mathbf{x}''\dot{\lambda}^2 - (\mathbf{J}^{-1}\mathbf{x}')^T \mathbf{B}(\mathbf{J}^{-1}\mathbf{x}')\dot{\lambda}^2\big], \quad (15)$$

where $\mathbf{J} = \frac{\partial \mathbf{k}}{\partial \mathbf{q}}$ is the manipulator's Jacobian matrix, $\mathbf{x}' = \frac{d\mathbf{x}}{d\lambda}$, $\mathbf{x}'' = \frac{d^2\mathbf{x}}{d\lambda^2}$, and $\mathbf{B}$ is the Hessian of the vector function $\mathbf{k}$ [7]. Equations (15) are substituted in (2) to yield the parametrized dynamics. In the redundant case considered here, $\mathbf{J}$ is no longer invertible, compromising the possibility of a unique parametrization. As a consequence, the pseudo-acceleration limits cannot be pre-computed: a state space grid may still be computed on the basis of the discrete values of the state, but the content of each single node should be reconsidered in light of this observation. Each node in the grid will then contain a state given by

$$\mathbf{s}_{l\mathbf{j}g}(i) = [\dot{\lambda}_l(i), \mathbf{q}_{\mathbf{j}g}(i)], \quad (16)$$

where $\mathbf{q}_{\mathbf{j}g}$ is the vector of joint positions obtained from (9) using the redundancy parameters $\mathbf{v_j}$ and selecting the g-th IK solution.

In what follows, for the sake of simplicity, whenever we do not need to refer to the specific position that a given state occupies in the grid, we drop the state's subscripts and refer to the generic state at stage i as $\mathbf{s}(i)$.

B. Constraints

In offline robotic planning, common constraints are limits on joint positions and their derivatives, actuation constraints, obstacles in the workspace, velocity and acceleration limits in the Cartesian space, initial/final joint configurations, cyclicity (for closed task space paths), power limits, maximum/minimum forces exchanged with the environment in an interaction scenario. Many other constraints can be designed depending on the specific application at hand. In the experiments that will follow in Section IV, we consider minimum/maximum joint positions, maximum joint velocities, accelerations, jerks, torques and torque rates [42].

Since the robot dynamics cannot be parametrized with respect to λ and its derivatives, the same holds for the constraints. Therefore, for redundant robots, they are directly verified on the joint variables. Hence, let us define the stage-dependent set $\mathcal{A}_i$, containing all nodes returning joint positions that respect the joint domains, the path constraint

and, possibly, imposed initial/final configurations. Such constraints are formalized as

$$\mathbf{s}(i) \in \mathcal{A}_i. \quad (17)$$

Joint velocity, acceleration and jerk limits can be directly encoded in equivalent stage-dependent, as well as state-dependent sets $\mathcal{B}_i^1$ $\mathcal{B}_i^2$ and $\mathcal{B}_i^3$, so that:

$$\dot{\mathbf{q}}(i) \in \mathcal{B}_i^1\big(\mathbf{s}(i)\big), \quad (18)$$
$$\ddot{\mathbf{q}}(i) \in \mathcal{B}_i^2\big(\mathbf{s}(i)\big), \quad (19)$$
$$\dddot{\mathbf{q}}(i) \in \mathcal{B}_i^3\big(\mathbf{s}(i)\big). \quad (20)$$

Equivalently, joint torque and torque rate limits can be encoded in similar sets $\mathcal{C}_i^1$ and $\mathcal{C}_i^2$:

$$\boldsymbol{\tau}(i) \in \mathcal{C}_i^1\big(\mathbf{s}(i)\big), \quad (21)$$
$$\dot{\boldsymbol{\tau}}(i) \in \mathcal{C}_i^2\big(\mathbf{s}(i)\big). \quad (22)$$

More commonly, all the quantities above are given within fixed domains that do not change along the trajectory, are not configuration dependent, and not velocity-dependent, but the generality of the framework allows for the accommodation of more complex constraints. For example, interacting torque constraints expressed as

$$\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_n]^T \in E(\mathbf{q}, \dot{\mathbf{q}}), \quad (23)$$

that cannot be parametrized [37], can be easily handled with DDP.

All the sets above can be combined together to define the set $\mathcal{D}_i$ of reachable states for a generic stage i, that is

$$\mathcal{D}_i = \mathcal{A}_i \cap \bigg\{ \mathbf{s}(i) : \dot{\mathbf{q}}(i) \in \mathcal{B}_i^1, \ddot{\mathbf{q}}(i) \in \mathcal{B}_i^2, \dddot{\mathbf{q}}(i) \in \mathcal{B}_i^3,$$
$$\boldsymbol{\tau}(i) \in \mathcal{C}_i^1, \dot{\boldsymbol{\tau}}(i) \in \mathcal{C}_i^2$$
$$\text{with } \mathbf{s}(i-1) \in \mathcal{A}_{i-1}, \ldots, \mathbf{s}(0) \in \mathcal{A}_0 \bigg\}. \quad (24)$$

C. Discrete Dynamic Programming Problem

Since all the states are available in the grid, the DDP problem is transformed into a graph search problem. This allows to adopt either a forward or a backward optimization scheme [36]. The objective function to optimize can be generally defined as

$$I(N_i) = \psi\big(\mathbf{s}(0)\big) + \sum_{k=1}^{N_i} \phi\big(\mathbf{s}(k-1), \mathbf{s}(k)\big), \quad (25)$$

where $\psi\big(\mathbf{s}(0)\big)$ is a cost associated to the initial state and ϕ is the cost computed locally between two adjacent states.

If, for instance, the objective function to minimize is the time, there is no cost associated to the initial state, i.e. $\psi\big(\mathbf{s}(0)\big) = 0$ and $I = t$. Assuming a backward Euler approximation, (25) becomes

$$t(N_i) = \sum_{k=1}^{N_i} \frac{\lambda(k) - \lambda(k-1)}{\dot{\lambda}(k)}. \quad (26)$$

A different approximation can be adopted at the beginning and at the end of the trajectory to cope with the case $\dot{\lambda} = 0$ [38].

To complete our formulation, we rewrite (26) in a recursive form and use the set $\mathcal{D}_i$ in (24), by minimizing over the admissible states:

$$t_{opt}(0) = 0,$$
$$t_{opt}(i) = \min_{\mathbf{s}(i) \in \mathcal{D}_i} \left[t_{opt}(i-1) + \frac{\lambda(i) - \lambda(i-1)}{\dot{\lambda}(i)} \right], \qquad (27)$$

where $t_{opt}(i)$ at a generic stage i is the *optimal return function* (or *value function*) and $t_{opt}(N_i)$ represents the optimal cost.

III. ALGORITHMIC IMPLEMENTATION

Based on the problem formulation above, we can define the time-optimal trajectory planning algorithm for redundant robots (TOTP-R) with DDP as in Algorithm 1.

Algorithm 1 DDP TOTP-R Algorithm

1: *Initialize state space grid through inverse kinematics and discretization of $\dot{\lambda}$, according to (9) and (16)*
2: *Initialize $\mathcal{A}_i$, $\forall i = 0..N_i$*
3: *Initialize $\mathcal{B}_i^1, \mathcal{B}_i^2, \mathcal{B}_i^3, \mathcal{C}_i^1, \mathcal{C}_i^2 \; \forall i = 0..(N_i - 1)$ with state-independent information*
4: *Initialize $\mathcal{D}_i = \varnothing, \forall i = 0..N_i$*
5: *Initialize cost map $t_{i,l,\mathbf{j},g} = +\infty \; \forall i, l, \mathbf{j}, g$*
6: $t_{0,l,\mathbf{j},g} \leftarrow 0 \; \forall l, \mathbf{j}, g$
7: $\mathcal{D}_0 \leftarrow \mathcal{A}_0$
8: **for** $i \leftarrow 0$ to $N_i - 1$ **do**
9:     **for each** $\mathbf{s}_{ljg} \in \mathcal{D}_i$ **do**
10:         **for each** $\mathbf{s}_{mkh} \in \mathcal{A}_{i+1}$ **do**
11:             *Compute $\dot{\mathbf{q}}, \ddot{\mathbf{q}}, \dddot{\mathbf{q}}, \boldsymbol{\tau}, \dot{\boldsymbol{\tau}}$*
12:             **if** Constraints in (18) - (22) are satisfied **then**
13:                 $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_{i+1} \cup \{\mathbf{s}_{mkh}\}$
14:                 *Compute instantaneous cost function ϕ*
15:                 **if** $t_{i,l,\mathbf{j},g} + \phi < t_{i+1,m,\mathbf{k},h}$ **then**
16:                     $t_{i+1,m,\mathbf{k},h} \leftarrow t_{i,l,\mathbf{j},g} + \phi$
17:                     *Let $\mathbf{s}_{ljg}$ at stage i be the predecessor of $\mathbf{s}_{mkh}$ at stage $i + 1$*
18: $t_{opt}(N_i) \leftarrow \min_{\mathbf{j},g} \left[t_{N_i,0,\mathbf{j},g} \right]$
19: *Build functions $\dot{\lambda}(i)$ and $\mathbf{q}(i)$ of optimal pseudo-velocities and joint positions by screening the predecessors map backwards*

The algorithm assumes that a task space path is given as a discrete set of points and that they are associated to discrete values of the curvilinear coordinate, from $\lambda = 0$ up to the length of the path $\lambda = L$. If this is not the case, but some analytic curve is given, it can be sampled at the desired rate. Although (1) is a uniform sampling, this is not a necessary condition for TOTP-R. At step 1, the redundancy parameters vector is discretized according to (14), which allows to compute IK solutions as in (9). The pseudo-velocity is also discretized according to (13), so that the grid nodes can be defined as in (16).

In agreement with [36] and [43], let us define the *extended aspect* as each connected set of joint configurations $\mathbf{q}$, such that the determinant of the augmented (with the redundancy parameters) kinematics' Jacobian $|\mathbf{J}_a(\mathbf{q})|$ never nullifies. On the basis of this definition, every combination of signs of the factors of $|\mathbf{J}_a(\mathbf{q})|$ identifies a different extended aspect [44],

achieving a partition of the configuration space. If these combinations are known to the IK solver in (9), IK solutions belonging to different extended aspects can be mapped to different grids. In this case, g is also an extended aspect selector and, if desired, can be fixed in (9) to avoid traversing singularities pertaining to the augmented Jacobian. However, in general, finding a proper factorization of $|\mathbf{J}_a(\mathbf{q})|$ is not trivial, in which case the aforementioned mapping cannot be established and the state space grids obtained through (9) are *non-homogeneous* [36], i.e. each grid possibly mixes solutions from different extended aspects. As a consequence, g cannot be used for limiting the manipulator's motion to specific sub-spaces of the configuration space.

At step 2, the nodes in the grid are enabled/disabled according to geometrical constraints including, for example, configurations that bring the robot to collide with the surrounding environment. Also, if IK solutions are homogeneous, entire grids can be excluded to make the DDP algorithm find a trajectory in specific extended aspects. The sets $\mathcal{A}_i$ can also be used to impose that the robot must start and end its motion at rest, or even stop along the path if this is required by the specific task. In time-optimal planning of non-redundant robots, the same sets $\mathcal{A}_i$ can be used to impose the maximum velocity curve (MVC) constraint [4], [5]. In this case, since we do not perform any parametrization of dynamics and constraints, this is not possible, and the MVC constraint is implicitly checked at the time of verifying (18)-(22) at step 12.

In most practical cases, constraints on joint velocities, accelerations, jerks, torques, and torque rates are given in terms of state-independent connected sets, e.g. $\dot{\mathbf{q}} \in [\dot{\mathbf{q}}_{min}, \dot{\mathbf{q}}_{max}]$. In this case, the sets in (18)-(22) are completely defined beforehand, as done at step 2. Conversely, if they were state-dependent, they could be re-computed at the time the constraints are checked, where the current velocity and acceleration of the system are known.

At a given stage i, for each pair of nodes (steps 9 and 10) the discrete-time functions (or sequences) of parameters $S_\lambda(i)$, $S_{\dot{\lambda}}(i)$, $S_\mathbf{v}(i)$, $S_g(i)$ are available through back-pointers to compute joint velocities, accelerations and torques as in (10)-(12) respectively, as well as joint jerks and torque rates with equivalent discrete approximations. If constraints are satisfied, we say that the node $\mathbf{s}_{mkh}(i + 1)$ can be reached by the node $\mathbf{s}_{ljg}(i)$. The former is then added to the set of nodes $\mathcal{D}_{i+1}$ that can be visited at the next stage (step 13). Among all the nodes at stage i that can reach $\mathbf{s}_{mkh}(i + 1)$, at step 17, we save the pointer to the one that provides the lowest cumulative cost (or optimal return function), according to (27).

After the process above has been repeated for all the task space points, the node with minimum cumulative traversing time is picked (step 18) and, from it, the information about the time law, i.e. $\dot{\lambda}(i)$, and exploitation of the null-space, i.e. $\dot{\mathbf{q}}(i)$, are retrieved by following the map of predecessors backwards (step 19). The timestamps can be computed through (6) and applied to the joint space path to obtain the optimal joint space trajectory. Likewise, optimal joint velocities, accelerations, and torques can be computed from (10)-(12).

A few considerations are worthwhile. The enforcement of higher-order derivative constraints, like (20) and (22), would in principle require the introduction of an additional axis corresponding to the pseudo-acceleration $\ddot{\lambda}$ [37], [45].

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

FERRENTINO et al.: DYNAMIC PROGRAMMING FRAMEWORK FOR OPTIMAL PLANNING OF REDUNDANT ROBOTS 7

TABLE I
PANDA LIMITS FROM JOINT 1 (TOP) TO JOINT 7 (BOTTOM)

q_{min}	q_{max}	$\dot{q}_{max}$	$\ddot{q}_{max}$	$\dddot{q}_{max}$	τ_{max}	$\dot{\tau}_{max}$
[rad]	[rad]	[rad/s]	[rad/s^2]	[rad/s^3]	[Nm]	[Nm/s]
-2.8973	2.8973	2.1750	15	7500	87	1000
-1.7628	1.7628	2.1750	7.5	3750	87	1000
-2.8973	2.8973	2.1750	10	5000	87	1000
-3.0718	-0.0698	2.1750	12.5	6250	87	1000
-2.8973	2.8973	2.6100	15	7500	12	1000
-0.0175	3.7525	2.6100	20	10000	12	1000
-2.8973	2.8973	2.6100	20	10000	12	1000

The employment of jerk and/or torque rate constraints becomes particularly important when the trajectories have to be executed on the real hardware, however the introduction of an additional dimension might make the problem intractable from the practical standpoint. This is the issue addressed in [39], where higher-order constraints are also enforced in a planning scenario based on dynamic programming, without redundancy. Therein, continuous and differentiable profiles are generated in the phase plane through interpolation between two consecutive stages. The constraints of interest, including jerk and torque rate, are verified at selected check points along such profiles. This way, the search space does not need to be augmented and the complexity can be controlled without exiting the phase plane. In this paper, we essentially adopt the same solution, but the type of interpolation is left as a user choice as an additional mean to control the complexity. In both cases, when higher-order constraints are checked without augmenting the search space, the global optimality might be compromised [39].

Furthermore, with respect to pure redundancy resolution at kinematic level and pure time-optimal planning for non-redundant robots, the TOTP-R search space is of an increased dimension. Practically, this rules out the usage of an arbitrarily fine discretization and more trade-offs should be considered. Therefore, our solutions might be, in some cases, far from the true global optimum, and we should rely on a more relaxed condition of *resolution optimality*. A proof of convergence of approximate dynamic programming, for the more generic case of multi-objective optimization, is provided in [46].

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

The TOTP-R algorithm is validated on the Panda 7-DOF manipulator by Franka Emika,[1] whose joint limits are reported in Table I.[2]

We rely on the dynamic model identified in [47] and related libraries. They allow access to the inertia matrix $\mathbf{H}(\mathbf{q})$, the Coriolis vector $\dot{\mathbf{q}}^T \mathbf{C}(\mathbf{q}) \dot{\mathbf{q}}$, the friction vector $\mathbf{d}(\dot{\mathbf{q}})$, and the gravity vector $\mathbf{g}(\mathbf{q})$ in (2) and (3). The friction model includes Coulomb terms and adopts a sigmoidal formulation to avoid discontinuities for low joint velocities.

The execution of the TOTP-R algorithm requires a set of parameters to be defined, that are crucial for the performance of the algorithm, both in terms of quality of the joint space

TABLE II
CUBIC SPLINE CONTROL POINTS FOR THE ELLIPSE-LIKE PATH

Coordinate in meters [m]	CP$_1$	CP$_2$	CP$_3$	CP$_4$	CP$_5$
x	0.5	0.5	0.5	0.5	0.5
y	0.0	-0.3	0.0	0.3	0.0
z	0.8	0.6	0.5	0.6	0.8

solution and CPU planning time. Indeed, this is common to other techniques (see, for example, [3]). They are:

- number of waypoints in the assigned task space path;
- pseudo-velocity limit to be used in the grid computation algorithm, which in principle should be large enough to contain the maximum pseudo-velocity that the phase plane trajectory can reach, that is not known beforehand;
- pseudo-velocity resolution, determining, on one side, the accuracy in the definition of the phase plane trajectory and, on the opposite, the computational complexity of the DDP algorithm;
- redundancy parameter resolution, determining, on one side, the capability of the redundant manipulator to exploit its null-space to optimize the performance index at hand and, on the opposite, such as before, the computational complexity of the algorithm.

The redundancy parameter is selected as the joint position q_4 (middle of the chain), because it simplifies the analytical inverse kinematics in (8) [48]. As far as the discrete approximations for derivatives are concerned, in our experiments, that are based on a single-threaded implementation, we select a simple backward Euler approximation so as to speed up the computation and consume less memory. However, with a high-performance implementation, more complex discrete approximations could be used and more accurate results should be expected.

B. Planning

Two paths are designed in the task space. One is a straight line path having $x = 0.5$ m, $z = 0.4$ m and y spanning for 0.5 m from $y_s = 0.25$ m to $y_e = -0.25$ m. The other is an ellipse-like line with $L = 1.45$, obtained through cubic splining of the control points (CP) in Table II. The end-effector orientation can be easily understood from the accompanying video.[3]

Both tasks constrain six dimensions, leaving one degree of freedom for redundancy resolution. In addition, the robot has to track the path as fast as possible and is free to select the IK solutions allowing for the shortest tracking time: no extended aspect is excluded beforehand and the robot can possibly visit more than one of them. Initial and final joint positions are not constrained and, therefore, they are part of the solution. The parameters of the algorithm are selected as a result of a trade-off between computation time and accuracy.

The TOTP-R algorithm is executed on the straight line path and the results for different sets of parameters are reported in Table III. PV stands for *pseudo-velocity*, RP for *redundancy parameter*. A necessary condition for global optimality is that at least one actuator saturates for each waypoint. In our DDP algorithm, saturation is never exact because of discretization, thus an actuator is considered in saturation if its velocity,

---

[1] https://www.franka.de/panda

[2] https://frankaemika.github.io/docs/

[3] https://t.ly/ZKelW

TABLE III
RESULTS OF THE TOTP-R ALGORITHM ON THE STRAIGHT LINE PATH

Plan ID	Number of waypoints	PV limit	RP res. (deg)	PV res.	Cost (s)
1	10	1.4	0.500	0.02	0.655
2	10	1.4	0.250	0.02	0.594
3	10	1.4	0.250	0.01	0.592
4	10	1.4	0.125	0.02	0.574
5	15	1.4	0.125	0.02	0.645
6	20	1.4	0.125	0.02	0.592

acceleration, jerk, torque or torque rate is above 95% of its capacity, in agreement with Table I. All solutions of Table III present a bang-bang behavior according to this approximation.

With reference to Table III, we note that the optimal cost consistently is in a neighborhood of 0.6 s, with a variability of a few milliseconds. By increasing the number of waypoints only (see plans 4-6), the cost may increase or decrease. We should remark that more waypoints correspond to a larger number of constraints in the task and less freedom in deviating from the true straight line. On the other hand, more waypoints allow to reduce the error related to the linear approximation that we make at each step by using a simple Euler integration scheme. Regardless of the number of waypoints selected by the user, we note that, since (9) is a positional IK scheme, the task space path output by the planner does not differ from the input one. Unlike other planning techniques, task deviations due to, e.g. integration of velocity or acceleration equations, are not possible. As expected, with a fixed number of waypoints, the cost decreases for finer resolutions of the redundancy parameter and the pseudo-velocity (see plans 1-4), although the improvement due to the latter, in our experiments, is negligible (see plans 2 and 3).

The actuation limits of the Panda robot (see Table I), in our experiments, are always such that constraints in (18)-(19) activate before constraints in (20)-(22). This is also due to the fact that planning is performed by considering zero load at the end-effector, while jerk limits are very large. The resolution-optimal joint space solution for plan 6 is reported in Fig. 3, its kinematic derivatives in Fig. 4, torques and their derivatives in Fig. 5. The computation time needed to find such a solution is 219 minutes on a 64-bit Ubuntu 18.04 LTS OS running on an Intel® Core™ i7-2600 CPU @ 3.40GHz ×8. The TOTP-R algorithm in Fig. 1 has been implemented in C++ over ROS[4] and MoveIt![5]. No multi-core execution model has been used in the experiments.

We note that, at first, joint 1's acceleration saturates to reach the maximum velocity, which persists for a large portion of the trajectory, between $t = 0.16$ s and $t = 0.37$ s. Here, there is a rather simultaneous saturation of joint 4's acceleration and joint 7's velocity, up until $t = 0.44$ s. In the last segment, joint 1's acceleration saturates again, followed by joint 2's acceleration to perform the final braking. For some segments of the trajectory, two actuators are in saturation at the same time, which is expected for a redundant manipulator, as argued in [9].

A representation of the null-space for the straight line path with 20 waypoints (plan 6) and joint position q_1 is reported in Fig. 6 with feasibility maps [36], [43]. Because of the

[4]https://www.ros.org/
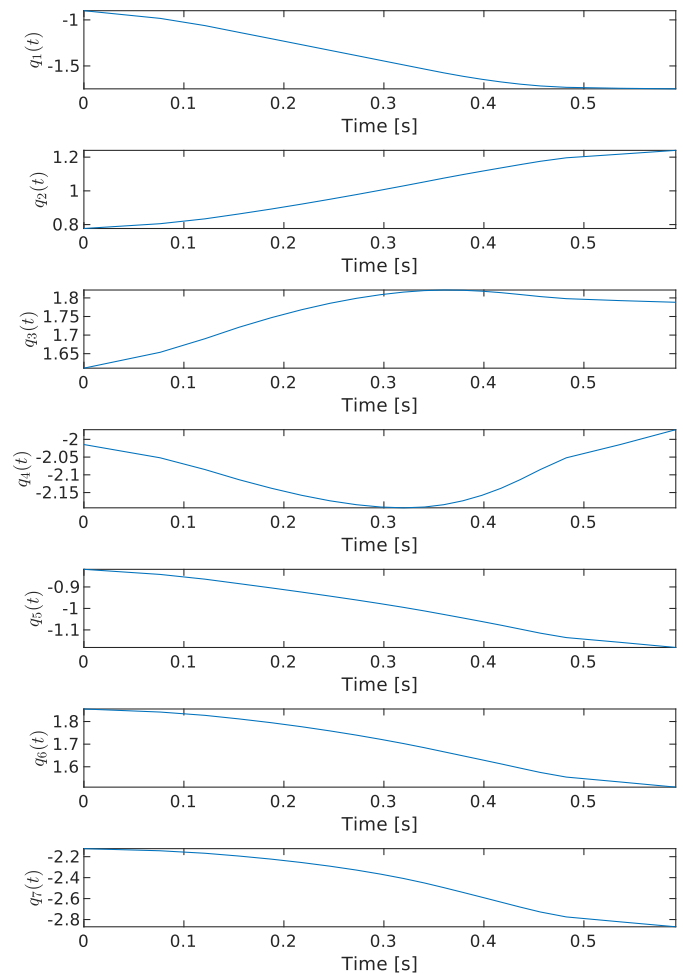[5]https://moveit.ros.org/

Fig. 3. Optimal joint positions for plan 6 of the straight line path.

task-manipulator geometry and joint limits, only 4 of the 8 grids expected for the Panda have feasible configurations, corresponding to the colored cells. The redundancy parameter, i.e. q_4, is represented along the y-axis in its physical domain (see Table I). The inverse kinematics solutions are returned by the kinematic solver *IKFast* [48] in a way that grids are not homogeneous.

The phase plane trajectory, that in the case of TOTP-R is a *phase space trajectory (PST)*, is reported in Fig. 7. As noted above, for redundant manipulators, the PST is a function of two independent variables that are λ (the progress along the path) and $\mathbf{v}$, the redundancy parameter(s). The 3D view provides an indication of how the DDP algorithm exploits the redundancy parameter to increase the pseudo-velocity and consequently compute a better solution with respect to a pre-assigned joint space path.

The best parameters for the straight line path are directly re-used for the ellipse-like path, to assess their consistency across different paths. Since the path is longer ($L = 1.45$ m), a higher number of waypoints should be used to keep a similar spatial resolution. The parameters used for this use case are reported in Table IV, together with the optimal cost. Graphs are omitted for the sake of brevity, but similar results are obtained. Even in this case, saturation only happens for velocities and accelerations.

In order to show that the algorithm is effectively imposing dynamic constraints, e.g. actuation torques, we plan the
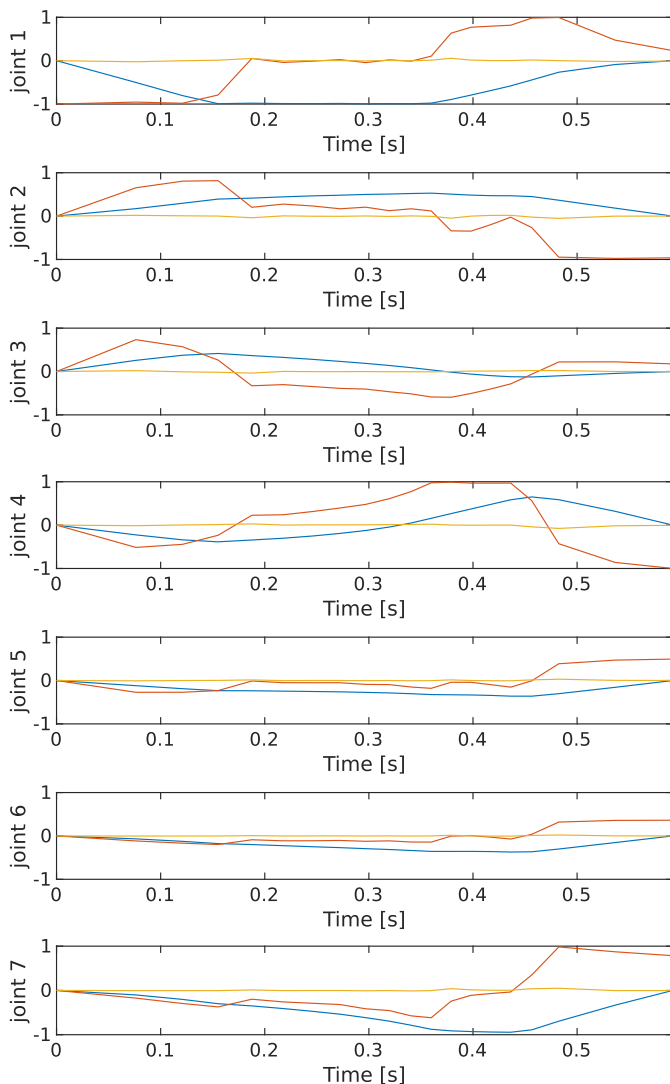
Fig. 4. Optimal joint velocities (blue), accelerations (red), and jerks (yellow) for plan 6 of the straight line path, normalized in $[-1, 1]$.

Fig. 5. Optimal joint torques (blue) and torque rates (red) for plan 6 of the straight line path, normalized in $[-1, 1]$.

TABLE IV

RESULTS OF THE TOTP-R ALGORITHM ON THE ELLIPSE-LIKE PATH WITH SAME PARAMETERS AS PLAN 6 OF TABLE III

Length (m)	Number of waypoints	PV limit	RP res. (deg)	PV res.	Cost (s)
1.45	60	1.4	0.125	0.02	1.856

straight line path again by setting stricter torque limits, i.e. $\tau_{max} = [15, 30, 87, 10, 12, 12, 12]^T$, and reusing the configuration parameters of plan 6 in Table III. Because of the stricter limits, the cost of the solution increases to 0.644 s. Fig. 8 shows the planned joint space trajectory, while Fig. 9 shows the saturating variables. In particular, joint 1's torque saturates first, until reaching the maximum velocity, which persists until $t = 0.43$ s. Here, joint 7's velocity saturates for a short period of time. In the last segment, we observe a rather simultaneous saturation of joint 1's and joint 2's torques to perform the final braking. Accelerations, jerks and torque rates never saturate.

C. Comparison With a Decoupled Approach

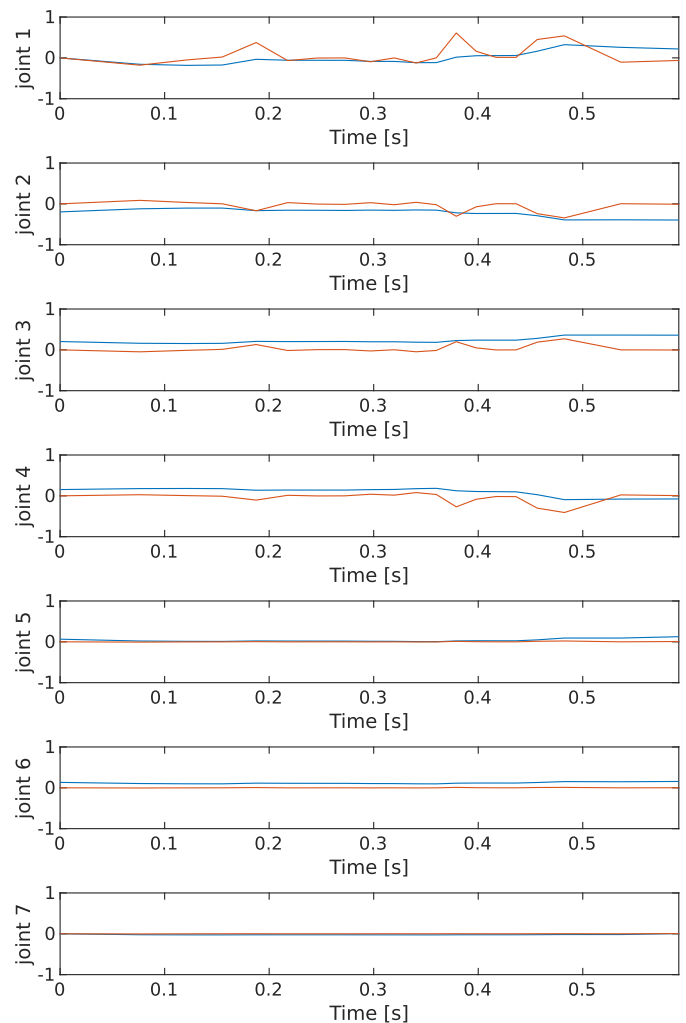Since we do not have the true minimum time trajectory available, we compare our DP algorithm with a decoupled approach from the literature, in order to assess the gain in performance given by the unified approach proposed in this paper. The same straight line and ellipse-like paths are inverted and time-parametrized with the algorithm in [1]. Each task space waypoint is inverted through the local Jacobian-based iterative optimization scheme.

$$\mathbf{q}_{k+1}(i) = \mathbf{q}_k(i) + \beta \mathbf{J}_k^\dagger \mathbf{e}_k - \alpha(\mathbf{I} - \mathbf{J}_k^\dagger \mathbf{J}_k)\nabla_{\mathbf{q}} c_k \quad (28)$$

where k is the iteration index, $\mathbf{J}^\dagger$ is the $n \times m$ full-rank Moore-Penrose pseudo-inverse of the manipulator's geometric Jacobian, $\mathbf{e}_k$ is the $m \times 1$ pseudo-twist Cartesian error vector, $\mathbf{I}$ is the identity matrix of appropriate dimension, $\nabla_{\mathbf{q}} c_k$ is the gradient of the cost function c selected for redundancy resolution, $\alpha = 10^{-4}$ and $\beta = 10^{-1}$ are gains on the null-space and Cartesian space correction terms, chosen after several trials. The cost is the projection of the dynamic manipulability ellipsoid onto the path's tangent $\mathbf{t}$, that is associated to the acceleration/deceleration capabilities of the arm along the path [1]:

$$c(\mathbf{q}) = \mathbf{t}^T (\mathbf{H}\mathbf{J}^\dagger)^T \mathbf{H}\mathbf{J}^\dagger \mathbf{t}. \quad (29)$$

After the joint-space path has been obtained, it is time-parametrized with the algorithm from [37], by using the same pseudo-velocity limit and a pseudo-velocity resolution of 0.01.
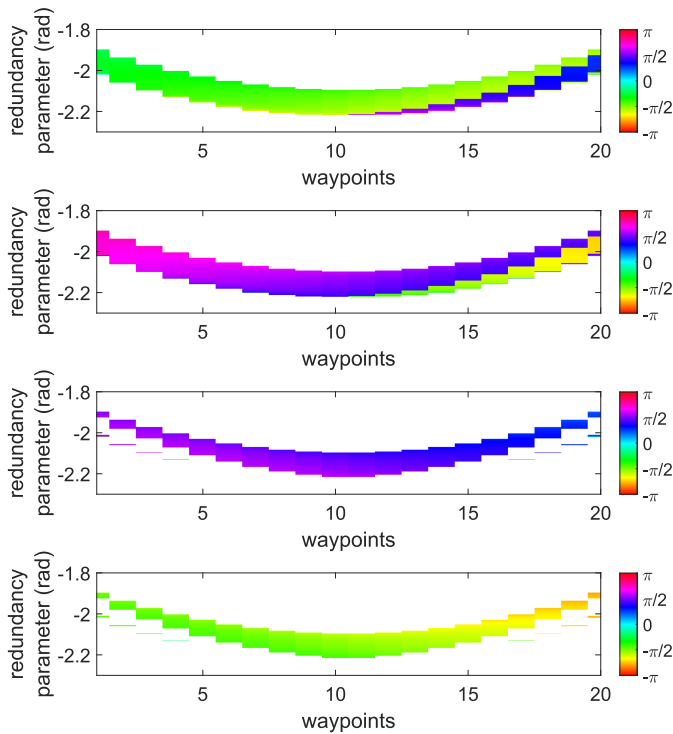
Fig. 6. Panda grids for $g = 1$ (top) to $g = 4$ (bottom), representing q_1 for the straight line path; q_1 color scale is in radians.
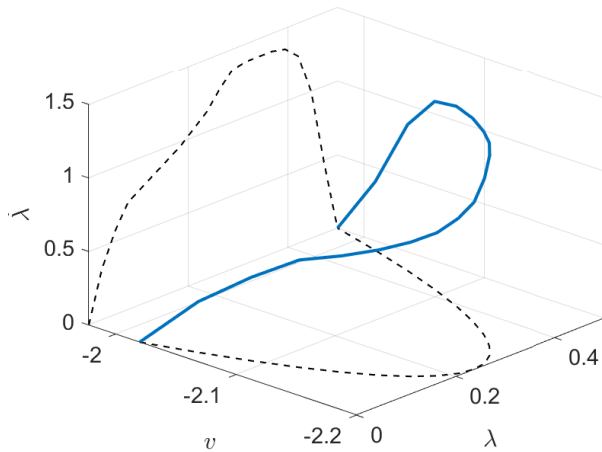

Fig. 7. Phase space trajectory (blue) with projections (black) on planes λ-$\dot{\lambda}$ and λ-v.
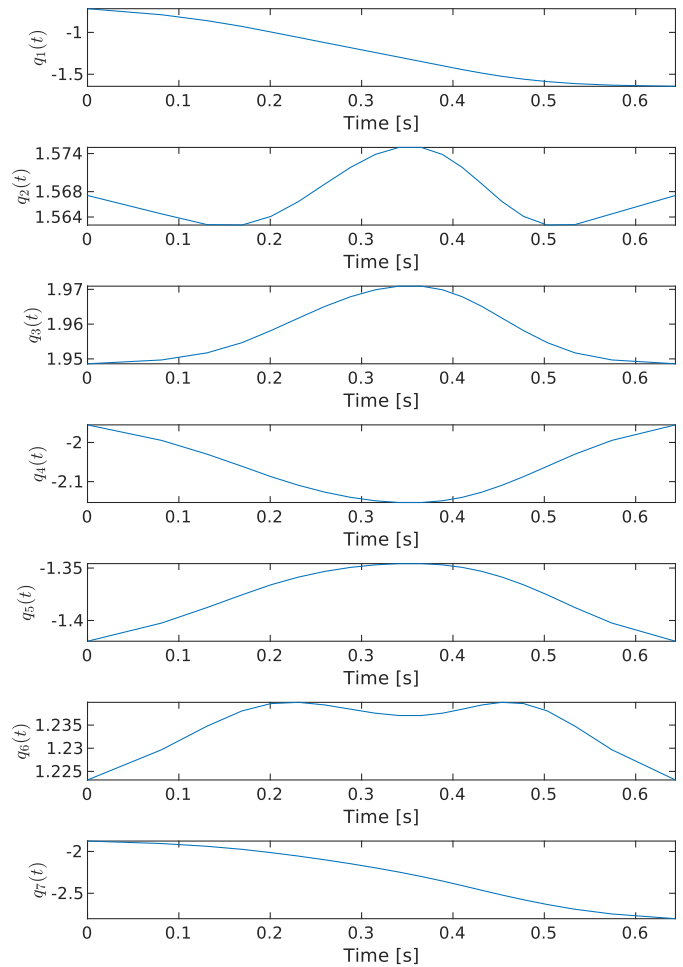

Fig. 8. Optimal joint positions for the straight line path, planned with the same parameters as plan 6, with stricter torque limits.

redundancy resolution drives some joints close to their limits, which forces the robot to slow down its progression on the path, reconfigure, and accelerate again. This is the main cause behind the huge deterioration of performance with respect to the unified approach.

D. Comparison With a Multiple Shooting Approach

In Section I-B, we identified the technique proposed in [14], [15], [16], and [17] as the state of the art in terms of time-optimal planning for redundant robots with a unified approach. We discussed some of its advantages, including the possibility of easily adding generic, possibly nonlinear constraints on the joint positions and their derivatives and generating smooth solutions. Here, we compare it with our approach on the straight line path. The problem formulation is:

$$\min_{\mathbf{y}} \; t_f$$
$$\text{s.t. } 0 \leq \lambda_k \leq 1$$
$$\dot{\lambda}_k \geq 0$$
$$\mathbf{q}_{min} \leq \mathbf{q}_k \leq \mathbf{q}_{max}$$
$$\dot{\mathbf{q}}_{min} \leq \dot{\mathbf{q}}_k \leq \dot{\mathbf{q}}_{max}$$
$$\ddot{\mathbf{q}}_{min} \leq \ddot{\mathbf{q}}_k \leq \ddot{\mathbf{q}}_{max}$$
$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_{max}$$

The trajectory tracking times for the straight line and the ellipse-like paths are 1.280 s and 2.557 s, about 113% and 38% higher than the unified approach. The CPU time for the whole planning process is about 1-2 minutes. Besides the mere assessment of cost and performance, we remark that the decoupled approach is very sensitive to the parameters α and β, as well as the specific objective function c selected for redundancy resolution. Other typical choices for c are joint displacement, velocity manipulability ellipsoid projection and multi-objective functions considering two or more performance indices together [49]. Furthermore, the initial joint positions must be assigned and cannot be retrieved as part of the solution. Lastly, the redundancy resolution procedure is not guarantee to converge in case of, e.g., strict joint limits, crossing of singularities. In our solution returning the cost of 1.280 s (more than twice the average time in Table III),

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

FERRENTINO et al.: DYNAMIC PROGRAMMING FRAMEWORK FOR OPTIMAL PLANNING OF REDUNDANT ROBOTS 11
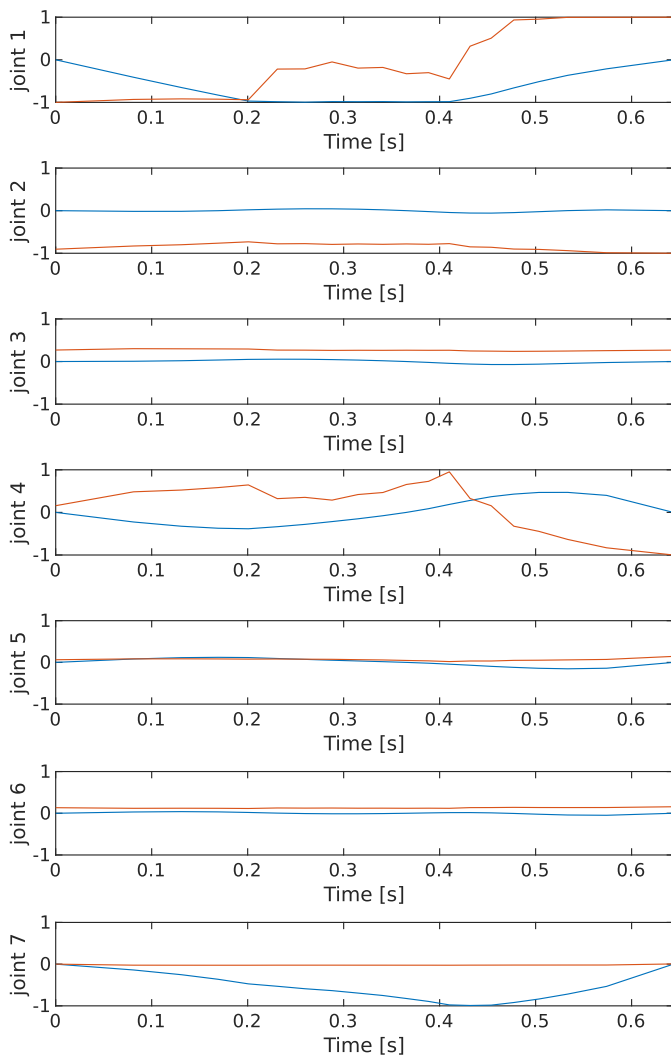

Fig. 9. Optimal joint velocities (blue) and torques (red), planned with the same parameters as plan 6, with stricter torque limits, normalized in $[-1, 1]$.

$$\ddot{\mathbf{q}}_k = \mathbf{J}_k^\dagger(\dot{\mathbf{w}}_k - \dot{\mathbf{J}}_k\dot{\mathbf{q}}_k) + \sum_{i=1}^{r} \gamma_{k,i}\mathbf{a}_{k,i}$$

$$\boldsymbol{\tau}_k = \mathbf{H}(\mathbf{q}_k)\ddot{\mathbf{q}}_k + \mathbf{f}(\mathbf{q}_k, \dot{\mathbf{q}}_k)$$

$$\mathbf{y}_{k+1} - \mathbf{g}_{RK4}(\mathbf{y}_k, \mathbf{u}_k) = \mathbf{0}$$

$$\mathbf{x}_k = \mathbf{k}(\mathbf{q}_k)$$

$$\lambda_0 = 0, \lambda_N = 1$$

$$\dot{\lambda}_0 = \dot{\lambda}_N = 0$$

$$\dot{\mathbf{q}}_0 = \dot{\mathbf{q}}_N = \mathbf{0} \qquad (30)$$

where N is the number of multiple shooting intervals, $k = 1, \ldots, N$, $\mathbf{y} = \left[\mathbf{y}_0^T, \mathbf{u}_0^T, \ldots, \mathbf{y}_{N-1}^T, \mathbf{u}_{N-1}^T, \mathbf{y}_N^T, t_f\right]^T$ is the decision vector, with $\mathbf{y}_k = \left[\lambda_k, \dot{\lambda}_k, \mathbf{q}_k^T, \dot{\mathbf{q}}_k^T\right]^T$, i.e. the state vector at the k-th interval, $\mathbf{u}_k = \left[\ddot{\lambda}_k, \gamma_{k,1}, \ldots, \gamma_{k,r}\right]^T$, i.e. the input vector at the k-th interval, $\mathbf{w}_k \in \mathbb{R}^m$ is the end-effector twist in the base frame, $\mathbf{a}_{k,i}$ are the r vectors in the null space basis of $\mathbf{J}_k$, $\gamma_{k,i}$ are their gains, and $\mathbf{g}_{RK4}$ is the 4-th order explicit Runge-Kutta integration scheme, employed to impose continuity between multiple shooting intervals.

We select $N = 20$, yielding 359 decision variables in $\mathbf{y}$, initialize $\mathbf{y}_k$ and $\ddot{\lambda}_k, \forall k$, to match the joint space trajectory obtained from Section IV-C, and we set $\gamma_k = \gamma_{k,1} = 0, \forall k$, as done in [14] and [15]. Since the algorithms in [14], [15], [16], and [17] are not publicly available, we implement problem (30) in MATLAB and solve it with fmincon using the interior-point method. The trajectory tracking time is 1.165 s (about 94% higher than our approach), the CPU time is 192 minutes. The involved quantities only partially saturate, which confirms the locally-optimal nature of the solution. Indeed, as noted in [14], [15], [16], and [17], global optimality is not guaranteed as problem (30) is non-convex, and neither is convergence for some choices of the parameters and initial guess.

E. Additional Comparison Trials

In order to confirm the correctness of the comparison, we ran additional trials assuming availability of a-priori information on the optimal solution of the straight line use case. In particular, by initializing problem (28) with the initial positions retrieved by our DDP algorithm (Fig. 3), setting c to joint displacement [49], and re-executing the steps of Section IV-C again, a trajectory tracking time of 0.628 s is achieved. If we use the latter solution as the initial guess for $\mathbf{y}$ in (30), the interior-point algorithm is able to consistently reach trajectory tracking times of ~ 0.600 s, for different configurations of the parameters N, γ_k. This confirms the strong dependency of the approaches in Sections IV-C and IV-D on the initial guess, as well as their locally-optimal nature. In our approach, no initial information on the solution needs to be available, as a global exploration of the search space is performed.

F. Execution

In this section, we assess the feasibility of the generated plan. We want to confirm that, despite the discrete approximations embedded in the DDP algorithm, the plan is actually executable through a standard interpolation and control scheme. We experimentally show that no additional requirements are imposed to the control system, besides controlling the plant in proximity of actuation limits.

The execution of the TOTP-R algorithm on either trajectory generates several control signals, as seen in Fig. 4 and Fig. 5. Before they can be used as commands for the Panda robot, it is necessary to interpolate them at the frequency of the controller, i.e. 1 kHz. We choose to control joint positions, while command torques are generated with an internal joint impedance controller, configured with maximum stiffness. Position control has already been used in other works, e.g. [3], dealing as well with execution of time-optimal joint space solutions, with satisfactory results. Furthermore, it is a typical choice as many robots do not allow for torque commands for practicality and safety reasons.

A high-level description of the control setup is shown in Fig. 10. As mentioned, Algorithm 1 is implemented in ROS/MoveIt! and deployed as a node. Task space paths $\mathbf{x}(\lambda)$ are provided in bag files and optimal joint space solutions, i.e. curves $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ are also stored in bag files. The controller manager of ros_control[6] is configured to work with the Panda robot hardware interface (provided through
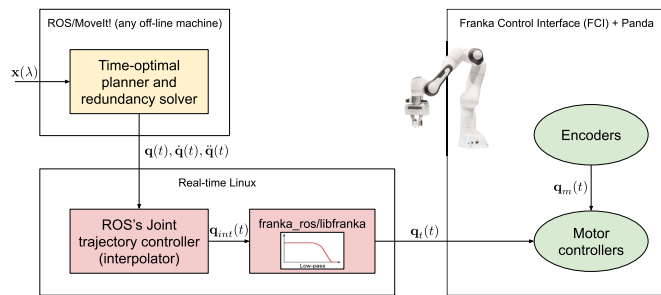
---

[6]http://wiki.ros.org/ros_control

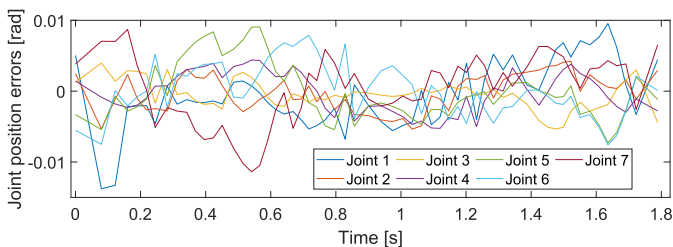Fig. 10.   Block diagram showing the control setup.

Fig. 11.   Joint position errors for the ellipse-like trajectory.

`franka_ros`[7]) and to load a robot-agnostic joint trajectory controller (JTC). Internally, the JTC performs quintic interpolation on the input signals to deliver smoother signals to the robot driver.

In order to guarantee that the interpolation of the planned trajectories still generates curves that are within bounds, we exploit some basic signal processing functions that are already provided by the manufacturer, i.e. a low-pass filter and a rate limiter. In particular, both of them are active to deliver less oscillating commands and ensure compliance to actuator limits. The employment of a filter to smooth the trajectory is also beneficial to eliminate the high frequency content that excites the unmodeled joint elasticity that characterizes joint transmissions in time-optimal control [50]. In fact, we note that controlling the motors in the proximity of their limits, so that they saturate, reserves no margin for the controller to compensate for unmodeled dynamics and/or uncertainties in the modeled dynamics. Some techniques explictly addressing this control issues can be found in [51], [52], [53], [54], [55], [56], [57], [58], [59], and [60].

The joint position tracking errors for the ellipse-like trajectory are reported in Fig. 11. The robot completes the motion in 1.859 s, 0.13% slower than planned with a maximum error in joint space of 0.8 deg. The delay introduced by the low-pass filter has been removed in the plots to provide a faithful comparison with the planned references. A video of the execution[8] is provided as supplemental material. While the actual tracking performance is dependant on the specific robot controller, we show that the trajectories planned with our methodology are feasible on a general-purpose architecture, i.e. the controller does not abort because of, e.g., out-of-bound or jerky references, poor smoothness or excessive trajectory tracking error. Moreover, the latter is comparable to other experiments of time-optimal trajectories execution [3].

---

[7]https://frankaemika.github.io/docs/franka_ros.html
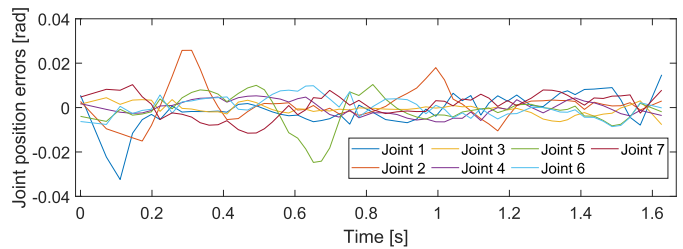[8]https://t.ly/ZKelW

Fig. 12.   Joint position errors for a trajectory 10% faster than the optimum.

In comparing with previous works, we should remark that our plans are calculated with the real robot capacities, as reported in Table I, while conservative values might be employed elsewhere, e.g. a given percentage of the actual limit.

The same planned trajectory is scaled in time to be 10% faster and is sent to the robot again to estimate its performance. In this case, the motion is completed in 1.70 s, 0.67% slower than planned, with a maximum error of 1.86 deg, associated to joint 1 at $t = 0.11$ s. Joint position tracking errors are reported in Fig. 12. As expected, the robot cannot be controlled beyond its capacities and the actual motion results in a remarkable degradation of tracking performances. We also note here that the controller aborts for trajectories that are more than 10% faster than the optimal plan.

V. CONCLUSION

In this paper, a DDP framework has been presented to plan trajectories for redundant robots along prescribed paths. The framework unifies redundancy resolution and trajectory planning so as to solve them together to effectively utilize kinematic redundancy to optimize the objective function defined at trajectory planning level. The extreme flexibility of DDP has been exploited to accommodate generic constraints that characterize real applications. DP ensures resolution-optimality, i.e. the asymptotic achievement of the globally-optimal solution as the dicretization step tends to zero.

Although it is perfectly suited to solve one or the other problem separately, and it is applicable to many offline planning scenarios, e.g. planning of cyclic operations, DDP still requires hours to days to solve a medium-complexity planning problem for redundant robots (one or two orders of magnitude higher than other approaches) and, most importantly, its scalability for systems characterized by a higher degree of redundancy, i.e. $r > 1$, is questionable. Also, in order to impose exact constraints on higher-order derivatives, the phase space should be further augmented with additional dimensions.

Nevertheless, the proposed state space formalization can be employed with optimization techniques other than DDP, such as randomized algorithms, for those scenario where planning time is a constraint. Additional future developments might regard a deeper study into decoupled techniques to evaluate whether some specific objective functions can be used for redundancy resolution (e.g., acceleration capability of the manipulator) that allow optimizing trajectory planning at a later stage.

ACKNOWLEDGMENT

REFERENCES

[1] P. Chiacchio, "Exploiting redundancy in minimum-time path following robot control," in *Proc. Amer. Control Conf.*, May 1990, pp. 2313–2318.

[2] F. Basile and P. Chiacchio, "A contribution to minimum-time task-space path-following problem for redundant manipulators," *Robotica*, vol. 21, no. 2, pp. 137–142, Feb. 2003.

[3] K. Al Khudir and A. De Luca, "Faster motion on Cartesian paths exploiting robot redundancy at the acceleration level," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3553–3560, Oct. 2018.

[4] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, Sep. 1985.

[5] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control*, vol. AC-30, no. 6, pp. 531–541, Jun. 1985.

[6] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE J. Robot. Autom.*, vol. RA-3, no. 2, pp. 115–123, Apr. 1987.

[7] J.-J. E. Slotine and H. S. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Trans. Robot. Autom.*, vol. 5, no. 1, pp. 118–124, Feb. 1989.

[8] Y. Chen, S. Y.-P. Chien, and A. A. Desrochers, "General structure of time-optimal control of robotic manipulators moving along prescribed paths," *Int. J. Control*, vol. 56, no. 4, pp. 767–782, Oct. 1992.

[9] M. Galicki and I. Pajak, "Optimal motions of redundant manipulators with state equality constraints," in *Proc. IEEE Int. Symp. Assem. Task Planning (ISATP)*, Jul. 1999, pp. 181–185.

[10] M. Galicki, "Time-optimal controls of kinematically redundant manipulators with geometric constraints," *IEEE Trans. Robot. Autom.*, vol. 16, no. 1, pp. 89–93, Feb. 2000.

[11] E. Ferrentino and P. Chiacchio, "Redundancy parametrization in globally-optimal inverse kinematics," in *Advances in Robot Kinematics*. Cham, Switzerland: Springer, Jun. 2018, pp. 47–55.

[12] A. Reiter, K. Springer, H. Gattringer, and A. Müller, "An explicit approach for time-optimal trajectory planning for kinematically redundant serial robots," *PAMM*, vol. 15, no. 1, pp. 67–68, Oct. 2015.

[13] S. Ma and M. Watanabe, "Time optimal path-tracking control of kinematically redundant manipulators," *JSME Int. J. Ser. C*, vol. 47, no. 2, pp. 582–590, 2004.

[14] A. Reiter, A. Müller, and H. Gattringer, "Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2016, pp. 6873–6878.

[15] A. Reiter, H. Gattringer, and A. Müller, "Redundancy resolution in minimum-time path tracking of robotic manipulators," in *Proc. 13th Int. Conf. Informat. Control, Autom. Robot.*, Jul. 2016, pp. 61–68.

[16] A. Reiter, H. Gattringer, and A. Müller, "On redundancy resolution in minimum-time trajectory planning of robotic manipulators along predefined end-effector paths," in *Informatics in Control, Automation and Robotics*. Cham, Switzerland: Springer, Nov. 2017, pp. 190–206.

[17] A. Reiter, A. Müller, and H. Gattringer, "On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1681–1690, Apr. 2018.

[18] X. Wang, Y. Gan, and X. Dai, "Time-optimal and smooth trajectory planning based on continuous pseudo-acceleration," in *Proc. Int. Conf. Adv. Robot. Mechatronics (ICARM)*, Jul. 2023, pp. 1059–1064.

[19] L. Wolinski and M. Wojtyra, "A novel QP-based kinematic redundancy resolution method with joint constraints satisfaction," *IEEE Access*, vol. 10, pp. 41023–41037, 2022.

[20] Y. Zhang and Y. Jia, "Motion planning of redundant dual-arm robots with multicriterion optimization," *IEEE Syst. J.*, vol. 17, no. 3, pp. 4189–4199, Sep. 2023.

[21] T. Marauli, H. Gattringer, and A. Müller, "Time-optimal path following for non-redundant serial manipulators using an adaptive path-discretization," *Robotica*, vol. 41, no. 6, pp. 1856–1871, Mar. 2023.

[22] J. Ogbemhe, K. Mpofu, and N. Tlale, "Optimal trajectory scheme for robotic welding along complex joints using a hybrid multi-objective genetic algorithm," *IEEE Access*, vol. 7, pp. 158753–158769, 2019.

[23] Z. Wang, Y. Li, K. Shuai, W. Zhu, B. Chen, and K. Chen, "Multi-objective trajectory planning method based on the improved elitist non-dominated sorting genetic algorithm," *Chin. J. Mech. Eng.*, vol. 35, no. 1, pp. 1–16, Feb. 2022.

[24] X. Zhang and G. Shi, "Multi-objective optimal trajectory planning for manipulators in the presence of obstacles," *Robotica*, vol. 40, no. 4, pp. 888–906, Jul. 2021.

[25] Z. Wang, Y. Li, P. Sun, Y. Luo, B. Chen, and W. Zhu, "A multi-objective approach for the trajectory planning of a 7-DOF serial–parallel hybrid humanoid arm," *Mechanism Mach. Theory*, vol. 165, Nov. 2021, Art. no. 104423.

[26] M. Schappler, "Pose optimization of task-redundant robots in second-order rest-to-rest motion with cascaded dynamic programming and nullspace projection," in *Informatics in Control, Automation and Robotics*. Cham, Switzerland: Springer, 2023, pp. 106–131.

[27] G. Singh and V. K. Banga, "Combinations of novel hybrid optimization algorithms-based trajectory planning analysis for an industrial robotic manipulators," *J. Field Robot.*, vol. 39, no. 5, pp. 650–674, Mar. 2022.

[28] J. Ma, Z. Cheng, X. Zhang, Z. Lin, F. L. Lewis, and T. H. Lee, "Local learning enabled iterative linear quadratic regulator for constrained trajectory planning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 5354–5365, Sep. 2023.

[29] G. Chen, M. Ma, S. Guo, B. Hou, and J. Wang, "An easy-implemented optimization method of trajectory planning based on polynomial interpolation," in *Proc. 40th Chin. Control Conf. (CCC)*, Jul. 2021, pp. 2802–2807.

[30] H. Wu, J. Yang, S. Huang, X. Ning, and Z. Zhang, "Multi-objective adaptive trajectory optimization for industrial robot based on acceleration continuity constraint," *Robot. Comput.-Integr. Manuf.*, vol. 84, Dec. 2023, Art. no. 102597.

[31] L. Žlajpah and A. Müller, "A task space decomposition algorithm for the inverse kinematics of functionally redundant manipulators," in *Proc. 20th Int. Conf. Adv. Robot. (ICAR)*, Dec. 2021, pp. 1048–1053.

[32] A. P. Pashkevich, A. B. Dolgui, and O. A. Chumakov, "Multiobjective optimization of robot motion for laser cutting applications," *Int. J. Comput. Integr. Manuf.*, vol. 17, no. 2, pp. 171–183, Mar. 2004.

[33] A. Dolgui and A. Pashkevich, "Manipulator motion planning for high-speed robotic laser cutting," *Int. J. Prod. Res.*, vol. 47, no. 20, pp. 5691–5715, Jul. 2009.

[34] A. Guigue, M. Ahmadi, R. Langlois, and M. J. Hayes, "Pareto optimality and multiobjective trajectory planning for a 7-DOF redundant manipulator," *IEEE Trans. Robot.*, vol. 26, no. 6, pp. 1094–1099, Dec. 2010.

[35] J. Gao, A. Pashkevich, and S. Caro, "Optimization of the robot and positioner motion in a redundant fiber placement workcell," *Mechanism Mach. Theory*, vol. 114, pp. 170–189, Aug. 2017.

[36] E. Ferrentino and P. Chiacchio, "On the optimal resolution of inverse kinematics for redundant manipulators using a topological analysis," *J. Mech. Robot.*, vol. 12, no. 3, Jun. 2020, Art. no. 031002.

[37] K. Shin and N. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Trans. Autom. Control*, vol. AC-31, no. 6, pp. 491–500, Jun. 1986.

[38] S. Singh and M. C. Leu, "Optimal trajectory generation for robotic manipulators using dynamic programming," *J. Dyn. Syst., Meas., Control*, vol. 109, no. 2, pp. 88–96, Jun. 1987.

[39] D. Kaserer, H. Gattringer, and A. Müller, "Nearly optimal path following with jerk and torque rate limits using dynamic programming," *IEEE Trans. Robot.*, vol. 35, no. 2, pp. 521–528, Apr. 2019.

[40] D. Kaserer, H. Gattringer, and A. Müller, "Time optimal motion planning and admittance control for cooperative grasping," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2216–2223, Apr. 2020.

[41] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[42] V. Petrone, E. Ferrentino, and P. Chiacchio, "Time-optimal trajectory planning with interaction with the environment," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10399–10405, Oct. 2022, doi: 10.1109/LRA.2022.3191813.

[43] J. A. Pámanes, P. Wenger, and J. L. Zapata, "Motion planning of redundant manipulators for specified trajectory tasks," in *Advances in Robot Kinematics*. Dordrecht, The Netherlands: Springer, 2002, pp. 203–212.

[44] P. Wenger, P. Chedmail, and F. Reynier, "A global analysis of following trajectories by redundant manipulators in the presence of obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, May 1993, pp. 901–906.

[45] H. Pham and Q.-C. Pham, "On the structure of the time-optimal path parameterization problem with third-order constraints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 679–686.

[46] A. Guigue, M. Ahmadi, M. J. D. Hayes, and R. G. Langlois, "A discrete dynamic programming approximation to the multiobjective deterministic finite horizon optimal control problem," *SIAM J. Control Optim.*, vol. 48, no. 4, pp. 2581–2599, Jan. 2009.

[47] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4147–4154, Oct. 2019.

[48] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2010.

[49] F. Storiale, E. Ferrentino, and P. Chiacchio, "Two-stage time-optimal planning of robots along pre-scribed paths with integral optimization of redundancy," in *Proc. 9th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Jul. 2023, pp. 1471–1476.

[50] J. Kim and E. A. Croft, "Online near time-optimal trajectory planning for industrial robots," *Robot. Comput.-Integr. Manuf.*, vol. 58, pp. 158–171, Aug. 2019.

[51] H. Arai, K. Tanie, and S. Tachi, "Path tracking control of a manipulator considering torque saturation," *IEEE Trans. Ind. Electron.*, vol. 41, no. 1, pp. 25–31, Feb. 1994.

[52] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. Robot. Autom.*, vol. 6, no. 5, pp. 554–561, Oct. 1990.

[53] Z. Shiller and H. Chang, "Trajectory preshaping for high-speed articulated systems," *J. Dyn. Syst., Meas., Control*, vol. 117, no. 3, pp. 304–310, Sep. 1995.

[54] Z. Shiller, H. Chang, and V. Wong, "The practical implementation of time-optimal control for robotic manipulators," *Robot. Comput.-Integr. Manuf.*, vol. 12, no. 1, pp. 29–39, Mar. 1996.

[55] J. Kieffer, A. J. Cahill, and M. R. James, "Robust and accurate time-optimal path-tracking control for robot manipulators," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 880–890, Dec. 1997.

[56] O. Dahl, "Path-constrained robot control with limited torques-experimental evaluation," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 658–669, Oct. 1994.

[57] J. Moreno-Valenzuela, "Tracking control of on-line time-scaled trajectories for robot manipulators under constrained torques," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA).*, May 2006, pp. 19–24.

[58] K. S. Eom, I. H. Suh, and W. K. Chung, "Disturbance observer based path tracking control of robot manipulator considering torque saturation," *Mechatronics*, vol. 11, no. 3, pp. 325–343, Apr. 2001.

[59] W. Niu and M. Tomizuka, "A new approach of coordinated motion control subjected to actuator saturation," *J. Dyn. Syst., Meas., Control*, vol. 123, no. 3, pp. 496–504, Jul. 1999.

[60] H. Y. Tam, "Minimum time closed-loop tracking of a specified path by robot," in *Proc. 29th IEEE Conf. Decis. Control*, Dec. 1990, pp. 3132–3137.

Heitor J. Savino (Member, IEEE) received the B.S. degree in mechatronics engineering from Amazon State University (UEA), the M.Eng. degree from Amazon Federal University (UFAM), and the Ph.D. degree in electrical engineering from the Federal University of Minas Gerais (UFMG) in 2016.

He was a Visiting Graduate Student with the Massachusetts Institute of Technology (MIT) and a member of the Interactive Robotics Group (IRG) (2015–2016). He conducted post-doctoral research with UFMG from 2016 to 2017 and LAAS-CNRS (2019–2020), as part of the H2020 PRO-ACT Project. He was an Assistant Professor with the Federal University of Alagoas (UFAL), Institute of Computing (IC) (2017–2021). Since 2022, he has been the Founder and a leading Researcher with the Ambev Robotics Laboratory, subsidiary of Anheuser-Busch InBev, to conduct research on robotics for the beverage industry.

Antonio Franchi (Fellow, IEEE) received the M.Sc. degree in electrical engineering and the Ph.D. degree in system engineering from the Sapienza University of Rome, Rome, Italy, in 2005 and 2010, respectively, and the H.D.R. degree in science from the National Polytechnic Institute of Toulouse in 2016. From 2010 to 2013, he was a Research Scientist with the Max Planck Institute for Biological Cybernetics, Tübingen, Germany. From 2014 to 2019, he was a tenured CNRS Researcher with LAAS-CNRS, Toulouse, France. From 2019 to 2021, he was an Associate Professor with the University of Twente, Enschede, The Netherlands, where he has been a Full Professor in aerial robotics control since 2022. Since 2023, he has been a Full Professor with the Department of Computer, Control and Management Engineering, Sapienza University of Rome. He has coauthored more than 160 peer-reviewed international publications on design and control of robotic systems applied to multi-robot systems and aerial robots.

Enrico Ferrentino (Member, IEEE) received the M.S. degree in computer and automation engineering from the Polytechnic University of Turin, Turin, Italy, and the Ph.D. degree in robotics from the University of Salerno, Fisciano, Italy. Between 2013 and 2014, he was a Visiting Scholar with the NASA Jet Propulsion Laboratory, Pasadena, USA, where he joined the Axel Tethered Robot Project with the Robotic Mobility Group, led by Dr. I. Nesnas. Between 2014 and 2017, he was a Ground Segment Engineer with ALTEC S.p.A., Turin, Italy, where he joined the ESA Robotic Mission ExoMars. In 2019, he was a Visiting Scholar with LAAS-CNRS, Toulouse, France, where he joined the H2020 PRO-ACT Project, in the team led by Prof. A. Franchi. Since 2020, he has been a Lecturer in robotics and medical robotics with the Department of Information and Electric Engineering and Applied Mathematics, University of Salerno, where he has been a Researcher with the Automatic Control Group since 2021. His research interests include optimal planning of redundant and cooperating robots, interaction control, human–robot collaboration, industrial, aerospace, and medical robotics.

Pasquale Chiacchio (Senior Member, IEEE) is currently a Professor in automatic control and robotics with the Department of Information and Electric Engineering and Applied Mathematics, University of Salerno. His main research interests include robotics and modeling and control of discrete event systems. In the robotics field, he has been working on robot control and identification, inverse kinematic problem, interaction control, control of redundant manipulators, and control of cooperative manipulators. In the discrete event systems field, he has been working on supervisory control based on monitors, optimal supervisory control, and formal specification for supervisory systems. The results have been published in the main journals of the sector and have been accompanied by an intense experimental activity. He has been a Coordinator of two Research Projects of National Interest (PRIN) and has been involved in research projects funded by the European Union (ECHORD, AIRobots, EuRoC, LOCOMACHS, and LABOR). He is a member of the Italian Society of Control Researchers. In December 2011, he was nominated Knight and then in July 2014, he was promoted to a Officer of the Order of Merit of the Italian Republic. Since 2016, he has been the Director of the Ph.D. Program in Information Engineering.