Repeatable Motion Planning for Redundant Robots Over Cyclic Tasks

Giuseppe Oriolo, *Fellow, IEEE*, Massimo Cefalo, and Marilena Vendittelli

***Abstract*—We consider the problem of repeatable motion planning for redundant robotic systems performing cyclic tasks in the presence of obstacles. For this open problem, we present a control-based randomized planner, which produces closed collision-free paths in configuration space and guarantees continuous satisfaction of the task constraints. The proposed algorithm, which relies on bidirectional search and loop closure in the task-constrained configuration space, is shown to be probabilistically complete. A modified version of the planner is also devised for the case in which configuration-space paths are required to be smooth. Finally, we present planning results in various scenarios involving both free-flying and nonholonomic robots to show the effectiveness of the proposed method.**

***Index Terms*— Motion planning, path planning for manipulators, redundant robots, repeatability.**

I. Introduction

KINEMATICALLY redundant robots [1] possess more degrees of freedom (dofs) than strictly needed to accomplish a given task and offer, therefore, increased dexterity and flexibility for pursuing additional objectives, among which the most important is arguably collision avoidance. Researchers have been studying kinematic redundancy since the 1980s, but only the last decade has witnessed a widespread adoption of redundant robotic systems, both in industrial and nonindustrial contexts. Popular examples include 7- or 8-dof fixed-base manipulators, bimanual robotic platforms, mobile manipulators, humanoid robots, and many others.

For fixed-base manipulators, standard tasks involve the end-effector, which should be placed at a certain position and/or orientation (e.g., for picking an object), or follow a certain path/trajectory (e.g., for cutting or welding). For mobile robots, tasks can be of a more general kind, e.g., navigating a corridor, tracking visual features, maintaining mechanical balance, and so on. Independently of the nature of the task variables, an important case is that of *cyclic* tasks, which are described by closed paths (cycles) in the task space. These are typically tasks that must be repeated over and over, a situation particularly relevant in industrial applications but of possible interest in other scenarios, e.g., service robotics.

Depending on the specific method used for its generation, the motion of the robot in the configuration space over a cyclic task may turn out to be closed or not. In the first case, the generated motion and the method itself are called *repeatable*. The repeatability property is obviously desirable under various aspects. The first is efficiency: if the method is repeatable, the configuration-space motion over subsequent task cycles will be a simple repetition of the first cycle motion, and therefore the computational effort is limited. The main advantage of repeatability, however, is that it improves *legibility* and *predictability* of the robot motion [2]. These properties are particularly valuable in any context that involves both humans and robots and can be exploited to make their coexistence and interaction intrinsically safer. On the other hand, lack of repeatability is clearly a drawback, because it means that the robot behavior is essentially unpredictable: the robot executes the cyclic task but assumes postures that are different from cycle to cycle. This obviously poses a threat also to the safety of the robot itself, because a motion that appears to be safe in early cycles may instead lead to collisions with obstacles in the long run.

A. Historical Perspective

Motion generation for redundant robots subject to task constraints is usually addressed through *kinematic control* [1], [3], sometimes called *redundancy resolution*. In this well-known approach, the robot is modeled as an elementary linear system (one simple integrator for each coordinate, the input being the corresponding velocity) with a nonlinear output function (the task). Feedback linearization is then used to achieve a linear and decoupled input-output map and a globally stable error dynamics. In view of the redundancy (more inputs than outputs), inversion of the input-output map requires a generalized inverse of the task Jacobian and results in an infinity of possible solutions for the velocity input, all expressed as an inverse solution plus an additional term, which does not perturb task satisfaction (*null-space velocity*). In control terms, this is a manifestation of the existence of a zero dynamics for the output function corresponding to the considered task.

Two alternatives are available for selecting at each instant one solution among the infinite candidates: either define a cost function and solve a local optimization problem, as originally proposed in [4], or perform a suitable task augmentation to obtain a "square" problem that admits a single solution [5]. Apart from some algorithmic aspects, these two options can be

Manuscript received December 8, 2016; revised May 8, 2017; accepted May 17, 2017. This paper was recommended for publication by Associate Editor P. Soueres and Editor A. Billard upon evaluation of the reviewers' comments. *(Corresponding author: Giuseppe Oriolo.)*

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Rome 00185, Italy (e-mail: oriolo@diag.uniroma1.it; cefalo@diag.uniroma1.it; venditt@diag.uniroma1.it)

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2017.2715348

shown to be largely equivalent, so we will focus on the first in the following discussion.

The safety requirement for robot motion may be taken into account at the optimization level, either by defining proximity to obstacles as a cost criterion or by adding specific inequality constraints related to collision avoidance in the workspace [6]. However, it is important to realize that—independently of the specific version—kinematic control is always a greedy strategy whose optimization capabilities are inherently local; as a consequence, it cannot guarantee completeness (finding a solution whenever one exists). In fact, during motion generation, it may happen that, depending on the previous history of the solution, the robot has reached a configuration where it cannot continue to execute the task without colliding with some obstacles or violating some other constraint (e.g., joint limits). From a general viewpoint, one may argue that the lack of backtracking capabilities makes kinematic control only suited for online or reactive motion generation, whereas in a full-information context it may prove severely suboptimal and even fail to solve relatively easy problems.

From the specific viewpoint of this paper, where we address both safety and repeatability, there exists a second, even more important drawback of kinematic control: the lack of repeatability. Consider, for example, its most popular version, i.e., *pseudoinverse control*; it is obtained by taking the Moore–Penrose pseudoinverse as a generalized inverse and setting the null-space velocity to zero, and corresponds to minimizing the norm of the generalized velocity. In spite of its popularity, this technique is not repeatable: this was first observed on simple planar robots by Klein and Huang [7], who pointed out how its application over a cyclic task leads to a drift in the configuration variables, whose final value is different from the initial value.

In 1988, Shamir and Yomdin [8] presented a necessary and sufficient condition (which was later refined in [9]) for repeatability of all kinematic control schemes in which the null-space velocity is set to zero (including pseudoinverse control). In particular, using differential geometric arguments, they proved that repeatability is obtained if and only if the distribution spanned by the columns of the chosen generalized inverse is involutive. This condition, which is very restrictive and therefore violated in most cases, was exploited to achieve *asymptotically repeatable* kinematic control in [10] and to create repeatable generalized inverses in [11]. Other works on the design of repeatable kinematic control schemes include, e.g., [12] and, more recently, [13], [14].

In any case, kinematic control schemes based on repeatable generalized inverses leave no room for optimization, in the sense that adding null-space velocities to the solution automatically destroys involutivity and therefore repeatability [10]. In other words, using repeatable generalized inverses amounts to losing most of the benefits of kinematic redundancy.

The extended Jacobian (EJ) technique, proposed by Baillieul [15] and revisited by Chang [16], deserves a special mention in this discussion. In a nutshell, this kinematic control method combines optimization with task augmentation: once a cost criterion has been chosen, the original task is augmented with additional constraints derived from the constrained optimality conditions. This leads to a square problem with a single velocity solution and makes the EJ method repeatable by construction. However, its application is impractical due to several difficulties, the most relevant being the appearance of *algorithmic singularities*, where the additional constraints come into conflict with the task. In the context of the present discussion, this is a manifestation of the *local* nature of the EJ and indicates that it is not possible to guarantee simultaneously repeatability and collision avoidance with this method.

B. Proposed Approach

Our approach to the considered problem proceeds from the observation that cyclic tasks are usually assigned and specified in advance in full-information contexts. We argue, therefore, that motion planning—rather than kinematic control—is the most appropriate setting for generating safe repeatable paths in the configuration space in the presence of this kind of tasks.

Indeed, within the vast motion planning literature, there exist methods that address the Task-Constrained Motion Planning (TCMP) problem, i.e., finding robot motions that are collision-free and simultaneously realize task constraints; see, for example, [17]–[24]. These methods, however, do not address the Repeatable-TCMP (R-TCMP) problem: if the assigned task path is closed, the resulting motion in the configuration space will not be repeatable in general.

In [25], we presented a preliminary study of a randomized planner targeting the R-TCMP problem (see the next section for a formal definition of this problem). The basic tool was a specialized motion generation scheme, first introduced in [20], which guarantees continued satisfaction of the task constraint throughout the motion. To achieve repeatability, this scheme was used within a bidirectional search: two exploration trees were grown in the task-constrained configuration space, the first in the forward direction and the second in the backward direction. When the two trees become sufficiently near, an attempt was made to connect them using a loop closure procedure designed using feedback control techniques.

This paper fully develops the ideas in [25]. In particular, it adds the following contributions:

1) a general loop closure procedure;
2) a theoretical investigation of the associated convergence conditions;
3) an extension to robotic systems subject to nonholonomic constraints;
4) a proof of probabilistic completeness;
5) a modified planner that generates smooth paths in the configuration space;
6) numerical results in new scenarios, including cases of mobile manipulation.

This paper is organized as follows. In the next section, the considered planning problem is formulated by specifying the underlying kinematic model and discussing the nature of the task assigned to the robot. Section III describes in detail the structure of the proposed planning method. The loop closure procedure, which is a key component of our planner, is thoroughly discussed in Section IV. A proof of probabilistic completeness is given in Section V. A modified version of the planner that is guaranteed to produce smooth paths in the configuration space is described

in Section VI. Numerical results for three different scenarios are presented in Section VII, while Section VIII offers some concluding remarks.

II. Problem Formulation

In this section, we first present the kinematic model used for generating motion of the considered robotic systems. Then, the nature of the assigned task is discussed. Finally, based on this preliminary material, the planning problem is defined in formal terms.

A. Motion Model

Consider a general robotic system whose configuration $\boldsymbol{q}$ takes values in an n-dimensional configuration space $\mathcal{C}$. By *general*, we mean that the robot may be subject to nonholonomic[1] constraints, whose number is denoted by r. The number of dofs (i.e., independent infinitesimal motions) is, therefore, $p = n - r$; if $r = 0$, the robot is actually *free-flying* in $\mathcal{C}$.

It is also assumed that the nonholonomic constraints involve only a subset[2] of the configuration variables. In particular, reorder and partition the configuration vector $\boldsymbol{q}$ as $(\boldsymbol{q}_c, \boldsymbol{q}_u)$, where the subvector of (nonholonomically) *constrained* coordinates $\boldsymbol{q}_c$ is n_c-dimensional and the subvector of *unconstrained* coordinates $\boldsymbol{q}_u$ is n_u-dimensional, with $n_u = n - n_c$. The r nonholonomic constraints are in Pfaffian form and act on the $\boldsymbol{q}_c$ subvector:

$$\boldsymbol{A}(\boldsymbol{q}_c)\dot{\boldsymbol{q}}_c = \boldsymbol{0}, \tag{1}$$

with the constraint matrix $\boldsymbol{A}$ full rank at any configuration $\boldsymbol{q} \in \mathcal{C} = \mathcal{C}_c \times \mathcal{C}_u$.

Since our motion planning method can be used for planning either paths or trajectories in $\mathcal{C}$, we shall consider a more general version of (1), where the time variable t is replaced by a generic path parameter s, with $s = t$ as a particular case:

$$\boldsymbol{A}(\boldsymbol{q}_c)\boldsymbol{q}_c' = \boldsymbol{0}, \tag{2}$$

with the notation $(\,)' = d(\,)/ds$ representing a derivative with respect to s.

In light of the above arguments, the motion model used for planning characterizes the admissible tangent vectors to the configuration-space path as

$$\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{q}_c' \\ \boldsymbol{q}_u' \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q}_c)\tilde{\boldsymbol{v}}_c \\ \tilde{\boldsymbol{v}}_u \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q}_c) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \tilde{\boldsymbol{v}}, \tag{3}$$

where the columns of the $n_c \times (n_c - r)$ matrix $\boldsymbol{G}(\boldsymbol{q}_c)$ are a basis for the null space of $\boldsymbol{A}(\boldsymbol{q}_c)$. The input vectors $\tilde{\boldsymbol{v}}_c$ and $\tilde{\boldsymbol{v}}_u$ are $(n_c - r)$-dimensional and n_u-dimensional, respectively, and they are collected in the p-dimensional vector $\tilde{\boldsymbol{v}} = (\tilde{\boldsymbol{v}}_c, \tilde{\boldsymbol{v}}_u)$. The tilde over these symbols indicates that in general ($s \neq t$) they are *geometric* inputs, rather than velocity inputs.

[1]Throughout this paper, it is assumed that holonomic constraints, if present, have been either solved (through appropriate choice of the generalized coordinates) or incorporated in the task definition.

[2]The developments in the paper are still valid if the nonholonomic constraints involve *all* the configuration variables. However, in this case the considered robotic system does not contain articulated limbs, and the repeatable motion planning problem for such a system is of little interest.

The kinematic model (3) applies to a large number of robotic systems, including fixed-based manipulators ($r = 0$) and nonholonomic mobile manipulators ($r > 0$). Its interpretation is clear: the geometric motion of the $\boldsymbol{q}_c$ coordinates in $\mathcal{C}_c$ is locally constrained by (2), whereas the $\boldsymbol{q}_u$ coordinates can move freely in $\mathcal{C}_u$. In any case, system (3) is fully controllable in view of the nonholonomy of the constraints (2).

B. Task Kinematics

The task assigned to the robot is described in terms of a set of coordinates $\boldsymbol{y}$ taking values in an m-dimensional space $\mathcal{T}$. Typical task coordinates may describe quantities related to manipulation (end-effector position and/or orientation), navigation (position of a representative point of the robot, e.g., the center of mass), or perception (placement of sensors in the workspace or directly of features in sensing space, as in visual servoing). In any case, the relationship between the task and the configuration coordinates is given by a direct kinematic map

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}),$$

whose differential (with respect to s) version is

$$\boldsymbol{y}' = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{J}_c(\boldsymbol{q}) & \boldsymbol{J}_u(\boldsymbol{q}) \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_c' \\ \boldsymbol{q}_u' \end{pmatrix},$$

where $\boldsymbol{J}(\boldsymbol{q}) = \partial \boldsymbol{f}/\partial \boldsymbol{q}$. Using (3), we can derive a more useful differential map from the actual geometric inputs to the admissible tangent vectors in the task space

$$\boldsymbol{y}' = \begin{pmatrix} \boldsymbol{J}_c(\boldsymbol{q})\,\boldsymbol{G}(\boldsymbol{q}_c) & \boldsymbol{J}_u(\boldsymbol{q}) \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{v}}_c \\ \tilde{\boldsymbol{v}}_u \end{pmatrix} = \boldsymbol{J}_t(\boldsymbol{q})\tilde{\boldsymbol{v}}. \tag{4}$$

The $m \times p$ matrix $\boldsymbol{J}_t(\boldsymbol{q})$ is simply called *task Jacobian* in the following. However, if $r > 0$, some elements of $\boldsymbol{J}_t$ are not partial derivatives due to the embedded nonholonomic constraints [26].

In the presence of nonholonomic constraints, two kinds of redundancy can be defined.

1) *Static* redundancy occurs when $n > m$ (the number of configuration coordinates exceeds the task dimension).
2) *Kinematic* redundancy occurs when $p > m$ (the number of dofs exceeds the task dimension).

These two concepts collapse in the absence of nonholonomic constraints, i.e., when $p = n$; this is, for example, the case of fixed-base manipulators. In general, however, kinematic redundancy implies static redundancy, whereas the converse is not true; for example, an elementary mobile manipulator consisting of a unicycle with a rigidly attached gripper is statically but not kinematically redundant for planar positioning tasks. Simple static redundancy ($n > p = m$) is not beneficial for planning because, unlike kinematic redundancy, it does not allow self-motions (i.e., continuous motions of the joints that do not change the value of the task variables). For this reason, we shall focus on the case of kinematic redundancy.

If the robot is kinematically redundant with respect to the task, the inverse image $\bar{\boldsymbol{q}} = \boldsymbol{f}^{-1}(\bar{\boldsymbol{y}})$ of a point $\bar{\boldsymbol{y}}$ in $\mathcal{T}$ may be either 1) an $(n - m)$-dimensional subset of $\mathcal{C}$, consisting of one or more disjoint manifolds, or 2) a finite number of configurations [27]. Task points of the first group include *regular*

points and *coregular* points (also called avoidable singularities), whereas the second group consists only of *singular* points (also called unavoidable singularities).

C. R-TCMP Problem

Consider a general robotic system $\mathcal{R}$ whose kinematic model is expressed as (3). The robot moves in a *workspace* $\mathcal{W}$ (a subset of $I\!R^3$) populated by static obstacles. Denote by $\mathcal{O} \subset \mathcal{W}$ and $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$, respectively, the volume occupied by the obstacles and by the robot at configuration $\boldsymbol{q}$.

The robot must perform a task described in terms of coordinates $\boldsymbol{y}$, whose differential map from the available geometric inputs is given by (4). It is assumed that the robot is kinematically redundant with respect to the task, i.e., $p > m$.

In this paper, we are interested in the case in which the assigned task is *cyclic*, i.e., it is specified as a closed *path* $\boldsymbol{y}_d(s)$ in $\mathcal{T}$, with $s \in [s_{\rm ini}, s_{\rm fin}]$ and $\boldsymbol{y}_d(s_{\rm ini}) = \boldsymbol{y}_d(s_{\rm fin})$. If $s = t$, then a closed task *trajectory* is assigned. It is also assumed that $\boldsymbol{y}_d(s)$ is continuously differentiable with respect to s and, for the motion problem to be well-posed, that

$$\boldsymbol{y}_d(s) \in \mathcal{T}^* \quad \forall s \in [s_{\rm ini}, s_{\rm fin}],$$

where $\mathcal{T}^*$ is the *nonsingular task space*, defined as the subset of $\mathcal{T}$ made of regular and coregular points. The initial configuration $\boldsymbol{q}_{\rm ini}$ is assigned,[3] with $\boldsymbol{f}(\boldsymbol{q}_{\rm ini}) = \boldsymbol{y}_d(s_{\rm ini})$.

The Repeatable Task-Constrained Motion Planning problem consists finding a configuration-space path $\boldsymbol{q}(s), s \in [s_{\rm ini}, s_{\rm fin}]$, such that:

1) $\boldsymbol{y}(s) = \boldsymbol{f}(\boldsymbol{q}(s)) = \boldsymbol{y}_d(s), \ \forall s \in [s_{\rm ini}, s_{\rm fin}]$ (the desired task path is realized);
2) $\boldsymbol{q}(s_{\rm ini}) = \boldsymbol{q}_{\rm ini} = \boldsymbol{q}(s_{\rm fin})$ (the configuration-space path is closed, hence *repeatable*);
3) $\mathcal{R}(\boldsymbol{q}(s)) \cap \mathcal{O} = \emptyset, \ \forall s \in [s_{\rm ini}, s_{\rm fin}]$ (all collisions are avoided).

The planning space for the R-TCMP problem is

$$\mathcal{C}_{\rm task} = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{y}_d(s), \text{for some } s \in [s_{\rm ini}, s_{\rm fin}]\}.$$

The manifold $\mathcal{C}_{\rm task}$, called *task-constrained configuration space* in the following, has the natural structure of a *foliation*, with s as a parameter. In fact, by defining the generic *leaf* as

$$\mathcal{L}(s) = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{y}_d(s)\},$$

the task-constrained configuration space can be characterized as

$$\mathcal{C}_{\rm task} = \bigcup_{s \in [s_{\rm ini}, s_{\rm fin}]} \mathcal{L}(s).$$

Since the assigned task path is closed, it is $\mathcal{L}(s_{\rm ini}) = \mathcal{L}(s_{\rm fin})$.

The existence of a solution to the above problem obviously depends on the placement of the obstacles in the workspace. In particular, denoting by $\mathcal{C}_{\rm free}$ the set of all configurations in $\mathcal{C}$ that are not in collision, R-TCMP is solvable if and only if there exists a closed path in $\mathcal{C}_{\rm task} \cap \mathcal{C}_{\rm free}$ that goes through $\boldsymbol{q}_{\rm ini}$ and crosses all leaves of $\mathcal{C}_{\rm task}$.

Note that limits on the admissible values of the generalized coordinates $\boldsymbol{q}$ (e.g., joint limits) can be readily incorporated in the above formulation by adding them to the list of constraints. In the rest of this paper, we will consider directly this extended formulation.

[3] If needed, $\boldsymbol{q}_{\rm ini}$ can be preliminarily computed by inverse kinematics.

III. R-TCMP Planner

The proposed randomized planner performs a bidirectional search in the task-constrained configuration space $\mathcal{C}_{\rm task}$ (see Fig. 1). In short, two trees $\mathcal{T}_{\rm fw}$ and $\mathcal{T}_{\rm bw}$ are generated: both are rooted at $\boldsymbol{q}_{\rm ini}$, but $\mathcal{T}_{\rm fw}$ explores $\mathcal{C}_{\rm task}$ in the forward direction (i.e., the direction of increasing s) from $s_{\rm ini}$, whereas $\mathcal{T}_{\rm bw}$ moves in the backward direction from $s_{\rm fin}$. To guarantee continued satisfaction of the task constraint throughout the motion, each tree is extended using a control-based variation [20] of the basic RRT mechanism [28]. Once $\mathcal{T}_{\rm fw}$ and $\mathcal{T}_{\rm bw}$ become sufficiently close, a purposely designed loop-closing procedure is invoked to attempt joining them.

During tree extension, we make use of a sequence of N samples of the desired task path $\boldsymbol{y}_d(s)$, respectively taken at $\{s_1 = s_{\rm ini}, s_2, \ldots, s_{N-1}, s_N = s_{\rm fin}\}$. In the following, we shall use the shorthand notations $\boldsymbol{y}_{d,i} = \boldsymbol{y}_d(s_i)$ and $\mathcal{L}_i = \mathcal{L}(s_i)$ to denote, respectively, the task path sample associated with s_i and the corresponding leaf of $\mathcal{C}_{\rm task}$. Since the task is cyclic, we have $\boldsymbol{y}_{d,1} = \boldsymbol{y}_{d,N}$ and $\mathcal{L}_1 = \mathcal{L}_N$. Any node produced during the search will lie on some leaf $\mathcal{L}_i$, with $i = 1, \ldots, N$, while any arc connecting two nodes will go across an infinity of intermediate leaves of $\mathcal{C}_{\rm task}$.

In the rest of this section, we describe in detail the tree extension mechanism and, in particular, the motion generation scheme used for guaranteeing task constraint satisfaction along the arcs. The loop-closing procedure will be presented separately in the next section.

A. Tree Extension

Let us consider first the extension of the forward tree $T_{\rm fw}$, which is rooted at $\boldsymbol{q}_{\rm ini}$.

At each iteration, a value $s_{\rm rand}$ is chosen arbitrarily (not necessarily at the discrete samples s_i) in the open interval $(s_{\rm ini}, s_{\rm fin})$. Then, a configuration $\boldsymbol{q}_{\rm rand}$ in $\mathcal{C}_{\rm task}$ is randomly generated as one of the inverse kinematic solutions[4] corresponding to $\boldsymbol{y}_d(s_{\rm rand})$, that is, $\boldsymbol{q}_{\rm rand}$ is chosen such that $\boldsymbol{f}(\boldsymbol{q}_{\rm rand}) = \boldsymbol{y}_d(s_{\rm rand})$.

The tree is then searched to identify the node $\boldsymbol{q}_{\rm near}$ that is closest to $\boldsymbol{q}_{\rm rand}$. Let s_i be the parameter value associated with the leaf $\mathcal{L}_i$ on which $\boldsymbol{q}_{\rm near}$ is located. At this point, a constraint-aware randomized scheme (see Section III-B) is invoked to generate a collision-free subpath that starts from $\boldsymbol{q}_{\rm near} \in \mathcal{L}_i$, ends at $\boldsymbol{q}_{\rm new} \in \mathcal{L}_{i+1}$ and continuously satisfies the task constraint. In the case of success, the subpath and $\boldsymbol{q}_{\rm new}$ are added to $T_{\rm fw}$ as an arc and a node, respectively.

Extension of the backward tree $T_{\rm bw}$, also rooted at $\boldsymbol{q}_{\rm ini}$, is performed in an analogous way, the only difference being that motion generation proceeds backwards from $s_{\rm fin}$, i.e., over de-

[4] Generating $\boldsymbol{q}_{\rm rand}$ in $\mathcal{C}_{\rm task}$ is computationally more expensive than simply taking $\boldsymbol{q}_{\rm rand}$ in $\mathcal{C}$; however, it generally results in the accelerated convergence of the planner.

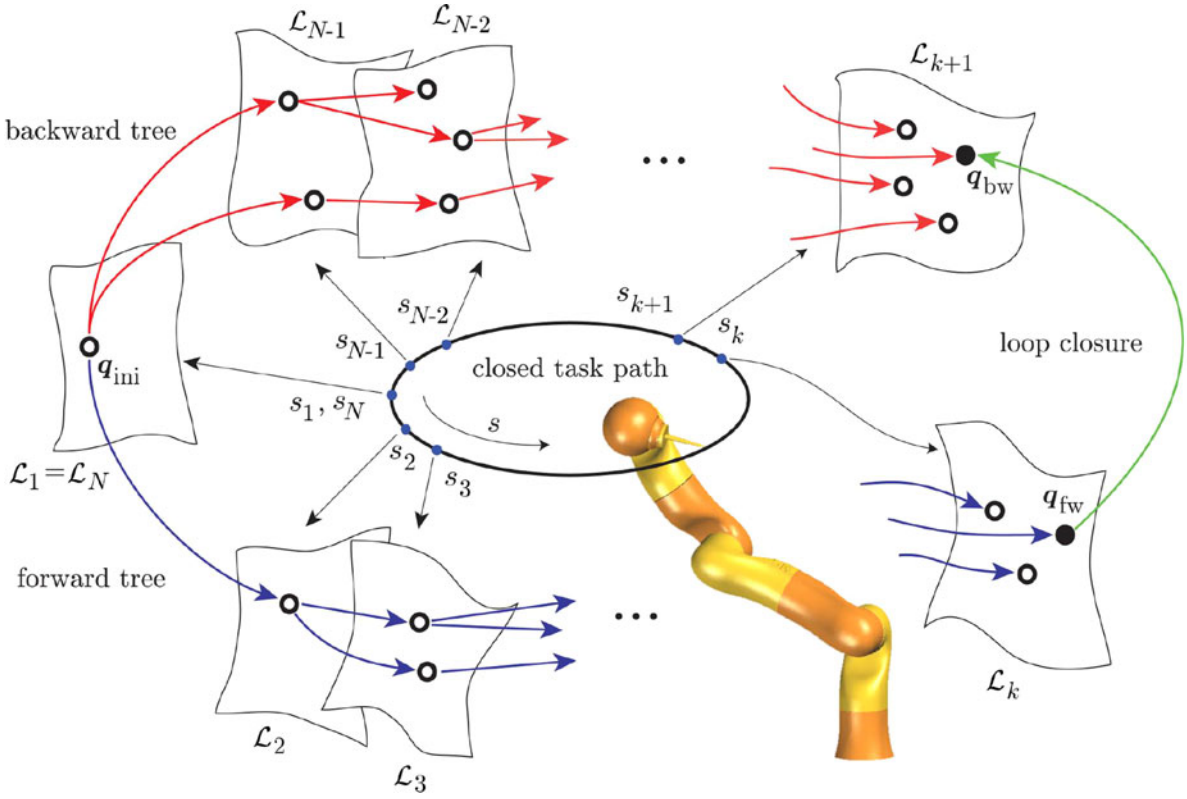

Fig. 1. The R-TCMP planner performs bidirectional search and loop closure in the task-constrained configuration space.

creasing values of s. Therefore, if $\boldsymbol{q}_{\rm near}$ lies on $\mathcal{L}_j$, the generated subpath will land on $\mathcal{L}_{j-1}$ (see Fig. 1).

The algorithm proceeds with a classical bidirectional search that balances exploration with exploitation. In the initial phase, the same $\boldsymbol{q}_{\rm rand}$ is used for extending both $T_{\rm fw}$ and $T_{\rm bw}$ independently. As the algorithm proceeds, extension steps are replaced with increasing frequency by connection steps aimed at reducing the gap between the two trees. To this end, the latest node added to one tree is used as $\boldsymbol{q}_{\rm rand}$ for the other.

Whenever a node is added to one of the two trees, the algorithm verifies whether the other tree has a node on an adjacent leaf. In this case, the bidirectional search is stopped and the loop-closing procedure is invoked (see Section IV). If the procedure is successful, a solution is extracted from $T_{\rm fw}$ and $T_{\rm bw}$; otherwise, tree extension is resumed.

B. Motion Generation

The motion generation scheme used in the proposed planner is constraint-aware, i.e., it complies with the constraint that the desired task path $\boldsymbol{y}_d(s)$ is realized for all values of s. In particular, tree arcs are subpaths in $\mathcal{C}_{\rm task}$ produced by the kinematic model (3) under the following input vector:

$$\tilde{\boldsymbol{v}} = \boldsymbol{J}_t^{\dagger}(\boldsymbol{q})(\boldsymbol{y}_d' + k_t\,\boldsymbol{e}_t) + (\boldsymbol{I} - \boldsymbol{J}_t^{\dagger}(\boldsymbol{q})\boldsymbol{J}_t(\boldsymbol{q}))\tilde{\boldsymbol{w}}, \qquad (5)$$

where:

1) $\boldsymbol{J}_t^{\dagger}$ is the pseudoinverse of the task Jacobian;
2) k_t is a positive scalar gain;
3) $\boldsymbol{e}_t = \boldsymbol{y}_d - \boldsymbol{y}$ is the task error;
4) $(\boldsymbol{I} - \boldsymbol{J}_t^{\dagger}\boldsymbol{J}_t)\tilde{\boldsymbol{w}}$ is the *null-space velocity*, obtained as the product of $(\boldsymbol{I} - \boldsymbol{J}_t^{\dagger}\boldsymbol{J}_t)$, the orthogonal projection matrix in the null space of $\boldsymbol{J}_t$, with $\tilde{\boldsymbol{w}}$, a p-dimensional *residual input* vector that can be chosen arbitrarily without affecting satisfaction of the task constraint.

Since by assumption $p > m$, if $\boldsymbol{J}_t$ is full rank we have $\boldsymbol{J}_t^{\dagger} = \boldsymbol{J}_t^T(\boldsymbol{J}_t\boldsymbol{J}_t^T)^{-1}$ and thus $\boldsymbol{J}_t\boldsymbol{J}_t^{\dagger} = \boldsymbol{I}$. Regardless of the choice of $\tilde{\boldsymbol{w}}$, plugging (5) in (4) gives $\boldsymbol{e}_t' + k_t\boldsymbol{e}_t = 0$. The task error system is then exponentially stable; since the initial error is zero ($\boldsymbol{f}(\boldsymbol{q}_{\rm ini}) = \boldsymbol{y}_d(s_{\rm ini})$), this means that, in principle, the desired task path will be tracked *exactly*.[5] In practice, exponential stability guarantees that even the drift associated with the numerical integration of (3)–(5) is reduced to a minimum.

When the forward tree $T_{\rm fw}$ is being extended, integration of (3)–(5) starts from the current $\boldsymbol{q}_{\rm near}$ for $s = s_i$ and stops at $s = s_{i+1}$. The residual input vector $\tilde{\boldsymbol{w}}$ is chosen randomly in a bounded subset W of $\mathbb{R}^n$ (this will be essential in the proof of probabilistic completeness) and kept constant[6] throughout the integration interval $[s_i, s_{i+1}]$. The result of the integration is a subpath in $\mathcal{C}_{\rm task}$ that goes from $\mathcal{L}_i$ to $\mathcal{L}_{i+1}$. Extension of the backward tree $T_{\rm bw}$ is achieved similarly, the only difference being that integration is performed backwards in s. Since system (3)–(5) is symmetric, any backward motion can be reversed to the direction in which s increases.

[5] This is unlike sampling-based constrained motion planners, in which the desired task is typically realized at the nodes (samples of $\mathcal{C}$) but violated along the arcs (subpaths joining the samples).

[6] As a consequence, the profile of $\tilde{\boldsymbol{w}}$ will be piecewise constant over $[s_1, s_N]$.

During the integration, the generated configuration-space path is continuously checked for 1) collisions (including self-collisions) between the robot and the obstacles, 2) violation of joint limits, and 3) rank deficiencies of $\boldsymbol{J}_t(\boldsymbol{q})$. If any of these conditions arises, generation of the current motion is aborted and control goes back to the main tree extension procedure.

Note that, in principle, one could use a different parameterization for the task space and configuration paths [20]. This would allow, for example, to perform a self-motion at the configuration level to modify the robot internal posture, while the task variables do not change. In this paper, we shall not exploit this opportunity and will avoid self-motions in order to produce smoother paths.

IV. Loop Closure

A fundamental component of our planner is the *loop closure* procedure, which takes a pair of configurations $\boldsymbol{q}_{\rm fw} \in \mathcal{L}_k$ and $\boldsymbol{q}_{\rm bw} \in \mathcal{L}_{k+1}$, respectively nodes of $\mathcal{T}_{\rm fw}$ and $\mathcal{T}_{\rm bw}$, and tries to join them with a path that avoids obstacles and simultaneously realizes the portion of task between the samples $\boldsymbol{y}_{d,k}$ and $\boldsymbol{y}_{d,k+1}$ (see Fig. 1). This amounts to performing a collision-free configuration transfer in an n-dimensional space while continuously satisfying an m-dimensional constraint. Therefore, we need to replace (3)–(5) with a constrained motion generation scheme that can reach an assigned final configuration. To this end, we take inspiration from the self-motion stabilization method of [29].

The basic idea is to choose a subset of $n-m$ generalized coordinates and to perform the desired reconfiguration on these. The motion of the remaining m coordinates is then computed so as to satisfy the task constraint. Under certain conditions, to be discussed in the following, these coordinates will converge to their desired values as well.

Let us partition vector $\boldsymbol{q}$ as $(\boldsymbol{q}_r, \boldsymbol{q}_b)$, where the subvector of *redundant* coordinates $\boldsymbol{q}_r$ is $(n-m)$-dimensional, and the subvector of *base* coordinates $\boldsymbol{q}_b$ is m-dimensional. The number of such possible partitions is $n!/m!(n-m)!$, i.e., the number of combinations (without repetition) of class m that can be drawn from n elements. Correspondingly, the $m \times p$ task Jacobian in (4) can be block-partitioned as $\boldsymbol{J}_t = (\boldsymbol{J}_{t,r} \;\; \boldsymbol{J}_{t,b})$, where $\boldsymbol{J}_{t,r}$ is $m \times (p-m)$ and $\boldsymbol{J}_{t,b}$ is $m \times m$.

A. Constrained Configuration Transfer via Feedback

Let us assume first that the robot is free-flying ($r = 0$, or $p = n$). The general motion model (3) reduces to

$$\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{q}'_r \\ \boldsymbol{q}'_b \end{pmatrix} = \tilde{\boldsymbol{v}}. \tag{6}$$

With the chosen partition, loop closure motion is generated by integrating (6) from $(\boldsymbol{q}_{\rm fw}, s_k)$ using the following input vector:

$$\tilde{\boldsymbol{v}} = \boldsymbol{H}_1(\boldsymbol{q})\tilde{\boldsymbol{u}} + \boldsymbol{H}_2(\boldsymbol{q})(\boldsymbol{y}'_d + k_t \boldsymbol{e}_t), \tag{7}$$

where

$$\boldsymbol{H}_1(\boldsymbol{q}) = \begin{pmatrix} \boldsymbol{I} \\ -\boldsymbol{J}_{t,b}^{-1}\boldsymbol{J}_{t,r} \end{pmatrix} \qquad \boldsymbol{H}_2(\boldsymbol{q}) = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{J}_{t,b}^{-1} \end{pmatrix}$$

and

$$\tilde{\boldsymbol{u}} = k_r(\boldsymbol{q}_{r,\rm bw} - \boldsymbol{q}_r). \tag{8}$$

As before, $\boldsymbol{y}'_d$ indicates the geometric derivative of the desired task path (here, the portion between $\boldsymbol{y}_{d,k}$ and $\boldsymbol{y}_{d,k+1}$), $\boldsymbol{e}_t$ is the task error, and k_t and k_r are positive gains. Note the structure of $\boldsymbol{H}_1$ and, in particular, of its lower $m \times (n-m)$ block, which guarantees that the task will be realized regardless of the choice of $\tilde{\boldsymbol{u}}$.

To analyze the convergence properties of the loop closure scheme, we need some preliminary definitions.

Given a partition $\boldsymbol{q} = (\boldsymbol{q}_r, \boldsymbol{q}_b)$, and the pair $(\boldsymbol{q}_{\rm fw}, \boldsymbol{q}_{\rm bw})$ of configurations to be joined, define an associated *degenerate robot* by letting $\boldsymbol{q}_r = \boldsymbol{q}_{r,\rm bw}$; in other words, the degenerate robot is obtained from the original robot by "freezing" the redundant coordinates at their final desired values. The degenerate robot is nonredundant, as its configuration vector reduces to $\boldsymbol{q}_b$ only, and therefore for a generic value of the task function $\boldsymbol{y}$ there exists only a finite number of inverse kinematic solutions. If the degenerate robot is *noncuspidal* (see, e.g., [30]), then any path that joins two such solutions must necessarily cross a singularity of $\boldsymbol{J}_{t,b}$. We say that two configurations of the degenerate robot belong to the same *homotopy class* if it is possible to go from one to the other without crossing a singularity of $\boldsymbol{J}_{t,b}$.

Proposition 1: Assume that the following conditions hold.

1) The degenerate robot is noncuspidal.
2) $\boldsymbol{q}_{b,\rm fw}$ and $\boldsymbol{q}_{b,\rm bw}$ belong to the same homotopy class for the degenerate robot.

Then, use of (7) and (8) from $(\boldsymbol{q}_{\rm fw}, s_k)$ produces a path that joins $\boldsymbol{q}_{\rm fw}$ to $\boldsymbol{q}_{\rm bw}$ (i.e., $\boldsymbol{q}_{r,\rm fw}$ to $\boldsymbol{q}_{r,\rm bw}$ and $\boldsymbol{q}_{b,\rm fw}$ to $\boldsymbol{q}_{b,\rm bw}$) and simultaneously satisfies the associated portion of task constraint, provided that $\boldsymbol{J}_{t,b}$ remains nonsingular throughout the integration.

Proof: Plugging (7) and (8) into (6) gives the following controlled dynamics:

$$\boldsymbol{q}'_r = \; k_r(\boldsymbol{q}_{r,\rm bw} - \boldsymbol{q}_r) \tag{9}$$

$$\boldsymbol{q}'_b = \; -\boldsymbol{J}_{t,b}^{-1}(\boldsymbol{q})\boldsymbol{J}_{t,r}(\boldsymbol{q})\tilde{\boldsymbol{u}} + \boldsymbol{J}_{t,b}^{-1}(\boldsymbol{q})(\boldsymbol{y}'_d + k_t\boldsymbol{e}_t). \tag{10}$$

The first equation implies that $\boldsymbol{q}_r$ converges (exponentially) to $\boldsymbol{q}_{r,\rm bw}$. Substituting (7) into (4), we obtain $\boldsymbol{e}'_t + k_t\, \boldsymbol{e}_t = 0$. This implies that $\boldsymbol{y}$ converges exponentially to the final desired task value $\boldsymbol{y}_d(\boldsymbol{q}_{\rm bw})$, and therefore $\boldsymbol{q}_b$ converges (exponentially) to one of the inverse kinematic solutions of the degenerate robot (i.e., the robot with $\boldsymbol{q}_r$ frozen at $\boldsymbol{q}_{r,\rm bw}$) corresponding to the final task value $\boldsymbol{y}_d(\boldsymbol{q}_{\rm bw})$.

Now, assume that the inverse kinematic solution to which $\boldsymbol{q}_b$ converges is $\boldsymbol{q}_{b,\rm fin} \neq \boldsymbol{q}_{b,\rm bw}$. Since the degenerate robot is noncuspidal, $\boldsymbol{q}_{b,\rm fin}$ must belong to a different class of homotopy than $\boldsymbol{q}_{b,\rm fw}$ and $\boldsymbol{q}_{b,\rm bw}$. Therefore, in moving from $\boldsymbol{q}_{b,\rm fw}$ to $\boldsymbol{q}_{b,\rm fin}$, the robot must have gone through a singularity of $\boldsymbol{J}_{t,b}$; but this would contradict the thesis. Hence, we can conclude that $\boldsymbol{q}_{b,\rm fin} = \boldsymbol{q}_{b,\rm bw}$. ■

Note the following points.

1) Integration of (6) under (7), (8) is possible as long as $\boldsymbol{J}_{t,b}$ remains nonsingular. This condition is impossible to check in advance because it depends on the desired task. Therefore, during the integration, the generated configuration-

space path must be continuously checked for singularity of $\boldsymbol{J}_{t,b}$. We will come back to this shortly when we will give the actual procedure for loop closure.

2) With the linear control law (8), $\boldsymbol{q}_r$ converges to $\boldsymbol{q}_{r,\mathrm{bw}}$ asymptotically in s. This means that at $s = s_{k+1}$, where the integration must be arrested, there will be a nonzero error. However, since convergence is exponential, the error can be made arbitrarily small by appropriately choosing k_r. An effective alternative is to replace (8) with a control law that provides finite-time convergence, such as

$$\tilde{\boldsymbol{u}} = k_r(\boldsymbol{q}_{r,\mathrm{bw}} - \boldsymbol{q}_r)^{\eta}, \tag{11}$$

where $\eta \in [0, 1)$, and exponentiation must be intended as componentwise and sign-preserving. In particular, denoting by q_r^i the ith component of the $(n-m)$-dimensional vector $\boldsymbol{q}_r$, the choice

$$k_r = \frac{\max\limits_{i=1,\ldots,n-m} \left(|q_{r,\mathrm{bw}}^i - q_{r,\mathrm{fw}}^i|\right)^{1-\eta}}{(1-\eta)(s_{k+1} - s_k)} \tag{12}$$

guarantees that $\boldsymbol{q}_r$ will converge to $\boldsymbol{q}_{r,\mathrm{bw}}$ exactly at $s = s_{k+1}$ (e.g., see [31]).

To conclude this discussion, we consider the case in which the robot is subject to nonholonomic constraints ($r > 0$, or $p < n$) so that the full motion model (3) applies. The proposed approach for task-constrained configuration transfer still works; however, an adaptation may be needed depending on the chosen partition. In fact, if the partition is such that $\boldsymbol{q}_r$ contains *all* the nonholonomically constrained coordinates $\boldsymbol{q}_c$, then (8) or (11) cannot be used as such, because the number of velocity inputs for those coordinates $(n_c - r)$ is less than their number (n_c). In this case, it is necessary to generate the components of $\tilde{\boldsymbol{u}}$ corresponding to the $\tilde{\boldsymbol{v}}_c$ using a *nonholonomic path planner*, i.e., a scheme[7] that can steer the $\boldsymbol{q}_c$ variables between two given values while complying with the nonholonomic constraints. For further discussion of loop closure in the presence of nonholonomic constraints, see the third planning scenario in Section VII, where we consider a nonholonomic mobile manipulator.

B. Example

Consider a 3R planar manipulator with unit link lengths and tip positioning as task (see Fig. 2). Define its configuration vector $\boldsymbol{q} = (q_1, q_2, q_3)$, where q_i is the ith relative joint angle, and consider the partition $\boldsymbol{q}_b = (q_1, q_2)$, $\boldsymbol{q}_r = q_3$. A simple computation shows that the singularities of $\boldsymbol{J}_{t,b}$ are met when $q_2 + q_3 = k\pi$, with $k = 0, \pm 1, \ldots$, i.e., when the tip of the manipulator falls on the first link axis.

Assume that a pair $(\boldsymbol{q}_{\mathrm{fw}}, \boldsymbol{q}_{\mathrm{bw}})$ is given. The degenerate robot is then obtained by freezing the third joint at a relative angle $\boldsymbol{q}_{r,\mathrm{bw}}$; this is equivalent to considering a two-link robot with a *virtual* second link replacing the second and third link of the original robot. Note that the degenerate robot is noncuspidal; in fact, for any generic value of the task variables (tip positioning), there exist only two inverse kinematic solutions, and they belong to different homotopy classes. These classes are easily visualized as elbow-up and elbow-down using the virtual link replacement (see Fig. 2).

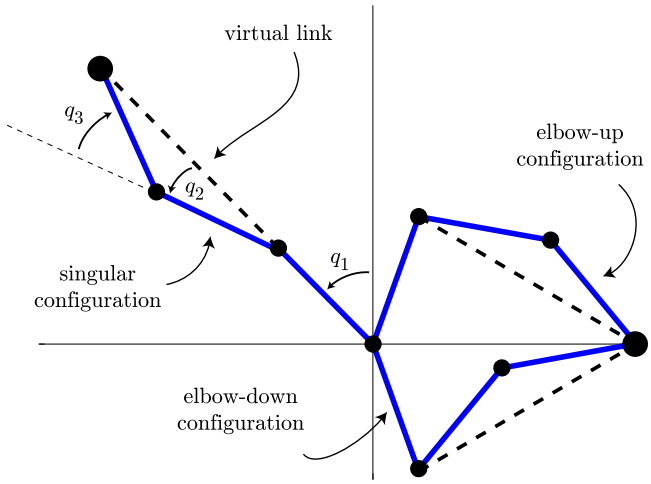

Fig. 2. Degenerate robot obtained from a 3R manipulator by freezing the third joint at a certain value: a singular configuration and two inverse kinematic solutions belonging to different homotopy classes (elbow-up/elbow down). Note the virtual link.

Proposition 1 applied to this 3R robot guarantees that loop closure will work between configurations belonging to the same homotopy class of the degenerate robot, provided that $\boldsymbol{J}_{t,b}$ remains nonsingular. This is confirmed by numerical simulations; for example, Fig. 3 shows two results obtained by applying (7), (8) from initial configurations belonging to different homotopy classes. As expected, in each case, the robot converges to the only inverse kinematic solution contained in the same class.

C. Procedure

Based on the previous arguments, we can now describe the actual loop closure procedure in detail. The data in input are $\boldsymbol{q}_{\mathrm{fw}} \in \mathcal{L}_k$, $\boldsymbol{q}_{\mathrm{bw}} \in \mathcal{L}_{k+1}$, and the associated portion of the desired task path $\boldsymbol{y}_d(s)$, $s \in [s_k, s_{k+1}]$.

First, all partitions $\boldsymbol{q} = (\boldsymbol{q}_r, \boldsymbol{q}_b)$ such that $\boldsymbol{J}_{t,b}$ is nonsingular at both $\boldsymbol{q}_{\mathrm{fw}}$ and $\boldsymbol{q}_{\mathrm{bw}}$ are put in a list, which is arranged in increasing order of cost. In our experiments, we used the distance $d(\boldsymbol{q}_{r,\mathrm{fw}}, \boldsymbol{q}_{r,\mathrm{bw}})$ as a cost function.[8] A partition is then chosen from the top of the list, and the conditions of Proposition 1 (noncuspidality of the associated degenerate robot and $\boldsymbol{q}_{b,\mathrm{fw}}$, $\boldsymbol{q}_{b,\mathrm{bw}}$ belonging to the same homotopy class) are checked. If they are satisfied, motion generation is started from $(\boldsymbol{q}_{\mathrm{fw}}, s_k)$ under (7), (8), or under (7), (11). During the integration, the generated configuration-space path is continuously checked. If either a singularity of $\boldsymbol{J}_{t,b}$ is met, or a collision occurs, or joint limits are violated, motion generation is aborted and the procedure restarts with the next partition of $\boldsymbol{q}$ in the list.

If loop closure is successful, a solution to the R-TCMP problem is reconstructed by patching together the configuration-space subpath from $\boldsymbol{q}_{\mathrm{ini}}$ to $\boldsymbol{q}_{\mathrm{fw}}$ on T_{fw}, the loop closure subpath

[7] Such schemes are available for a large class of nonholonomic robots. In particular, all differentially flat robots admit a nonholonomic path planner. See, for example, [32, Sec. 11.5.3].

[8] Other choices of cost function are possible; for example, one may use $1/\min(\det \boldsymbol{J}_{t,b}(\boldsymbol{q}_{\mathrm{fw}}), \det \boldsymbol{J}_{t,b}(\boldsymbol{q}_{\mathrm{bw}}))$ to characterize the proximity of $\boldsymbol{J}_{t,b}$ to singularities at both the start and goal configuration.

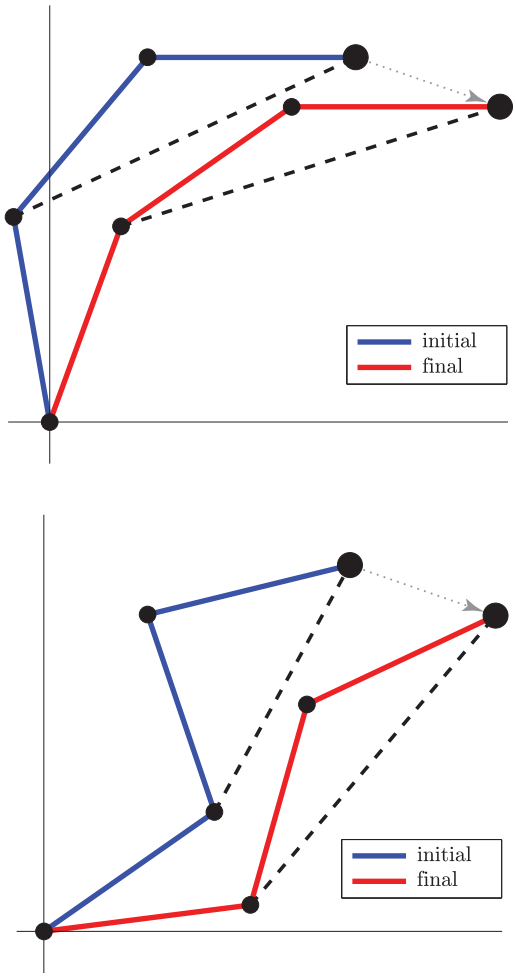

Fig. 3. Two results of the loop closure procedure for the same task path but initial configurations belonging to two different homotopy classes of the degenerate robot (above: elbow-up; below: elbow-down). In each case, the robot converges to the only inverse kinematic solution contained in the same class.

from $\boldsymbol{q}_{\rm fw}$ to $\boldsymbol{q}_{\rm bw}$, and the subpath from $\boldsymbol{q}_{\rm bw}$ to $\boldsymbol{q}_{\rm ini}$ on $T_{\rm bw}$ (in the reverse direction). If instead motion generation fails for any feasible partition of $\boldsymbol{q}$, the loop closure procedure reports a failure and control goes back to the planner, which will attempt further extension of the trees.

V. Probabilistic Completeness

We will now discuss the convergence properties of the proposed planner. In particular, we will show that the random choice of the residual input vector $\tilde{\boldsymbol{w}}$ in (5) is instrumental in achieving probabilistic completeness.

Proposition 2: Assume that a solution path $\boldsymbol{q}^*(s)$ exists which can be produced by the planner. Then, the probability of generating such path converges to 1 as the number of iterations increases.

Proof: By assumption, $\boldsymbol{q}^*(s)$, $s \in [s_1, s_N]$, is a path in $\mathcal{C}_{\rm task} \cap \mathcal{C}_{\rm free}$ obtained by joining 1) a subpath on the forward tree from $\boldsymbol{q}_{\rm ini} \in \mathcal{L}_1$ to $\boldsymbol{q}^*_{\rm fw} \in \mathcal{L}_k$, for some $k=2,\dots,N-2$; 2) a loop closure from $\boldsymbol{q}^*_{\rm fw} \in \mathcal{L}_k$ to $\boldsymbol{q}^*_{\rm bw} \in \mathcal{L}_{k+1}$; and 3) a (reversed) subpath on the backward tree from $\boldsymbol{q}^*_{\rm bw} \in \mathcal{L}_{k+1}$ to $\boldsymbol{q}^*_{\rm ini} \in \mathcal{L}_N = \mathcal{L}_0$ (see Fig. 1). In particular, the first and the third subpaths are obtained by integrating (3)–(5) with residual inputs $\tilde{\boldsymbol{w}}$ that are constant within each interval $[s_i, s_{i+1})$, with $i = 1,\dots,k-1$ for the first and $i = k+1,\dots,N-1$ for the third, whereas the second subpath comes from the integration of (6)–(7) from s_k to s_{k+1}. In view of Proposition 1, we can also assume that $\boldsymbol{q}^*_{\rm fw}$ and $\boldsymbol{q}^*_{\rm bw}$ belong to the same homotopy class of the degenerate robot.

Let us start by focusing on the second subpath. Since it is produced by integrating (6)–(7) from s_k to s_{k+1}, it is a continuous function of $\boldsymbol{q}^*_{\rm fw}$ (the initial condition of the integration) and $\boldsymbol{q}^*_{\rm bw}$ (which acts as a parameter via (8)). Hence, there exist two spherical neighborhoods, respectively, $\mathcal{S}(\boldsymbol{q}^*_{\rm fw})$ of $\boldsymbol{q}^*_{\rm fw}$ on $\mathcal{L}_k$ and $\mathcal{S}(\boldsymbol{q}^*_{\rm bw})$ of $\boldsymbol{q}^*_{\rm bw}$ on $\mathcal{L}_{k+1}$, such that for any pair of configurations $(\boldsymbol{q}_1, \boldsymbol{q}_2)$ with $\boldsymbol{q}_1 \in \mathcal{S}(\boldsymbol{q}^*_{\rm fw})$ and $\boldsymbol{q}_2 \in \mathcal{S}(\boldsymbol{q}^*_{\rm bw})$, the loop closure conditions still hold; i.e., $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ belong to the same homotopy class for the degenerate robots and $\boldsymbol{J}_{t,b}$ remains nonsingular throughout the integration.

The previous argument shows that to establish probabilistic completeness, it is sufficient to prove that the probability of generating a configuration in $\mathcal{S}(\boldsymbol{q}^*_{\rm fw}) \cap \mathcal{L}_k$ with the forward tree (first subpath) and in $\mathcal{S}(\boldsymbol{q}^*_{\rm bw}) \cap \mathcal{L}_{k+1}$ with the backward tree (third subpath) tends to 1. Clearly, it is enough to prove the first part only, as the planner behavior is symmetrical.

Let us consider then the first solution subpath

$$\boldsymbol{q}^*_{\to k} = \{\boldsymbol{q}^*(s), s \in [s_1, s_k]\},$$

which ends at $\boldsymbol{q}^*(s_k) = \boldsymbol{q}^*_{\rm fw}$ on $\mathcal{L}_k$, for some $k=2,\dots,n-2$, and let

$$\tilde{\boldsymbol{w}}^*_{\to k} = \{\tilde{\boldsymbol{w}}^*_1, \dots, \tilde{\boldsymbol{w}}^*_k\}$$

be the associated sequence of residual input vectors $\tilde{\boldsymbol{w}}$. Define the *clearance* of $\boldsymbol{q}^*_{\to k}$ as

$$\gamma = \min_{s \in [s_1, s_k]} d(\boldsymbol{q}^*(s)),$$

where $d(\boldsymbol{q})$ denotes the minimum distance between a configuration and the $\mathcal{C}$-obstacle region. At this point, we can define a circular tube in $\mathcal{C}$ around $\boldsymbol{q}^*_{\to k}$, denoted by $T(\boldsymbol{q}^*_{\to k})$, whose cross-section has radius $\rho = \min(\gamma, \beta)$, with β the radius of $\mathcal{S}(\boldsymbol{q}^*_{\rm fw})$. By construction, any path produced by the planner in $T(\boldsymbol{q}^*_{\to k})$ realizes the task, is safe, and ends at a configuration in $\mathcal{S}(\boldsymbol{q}^*_{\rm fw}) \cap \mathcal{L}_k$.

Once again, we can rely on the continuity of solutions of differential equations to claim that there exists a collection of spherical neighborhoods in $\mathbb{R}^n$

$$S(\tilde{\boldsymbol{w}}^*_{\to k}) = \{S(\tilde{\boldsymbol{w}}^*_1), \dots, S(\tilde{\boldsymbol{w}}^*_k)\}$$

such that any choice of $\tilde{\boldsymbol{w}}_{\to k}$ in $S(\tilde{\boldsymbol{w}}^*_{\to k})$ (i.e., of $\tilde{\boldsymbol{w}}_1$ in $S(\tilde{\boldsymbol{w}}^*_1)$, of $\tilde{\boldsymbol{w}}_2$ in $S(\tilde{\boldsymbol{w}}^*_2)$, and so on) produces a first subpath $\boldsymbol{q}_{\to k}$ in $T(\boldsymbol{q}^*_{\to k})$. Considering that the measure of $S(\tilde{\boldsymbol{w}}^*_{\to k})$ in $\mathbb{R}^n$ is nonzero, and that the residual inputs $\tilde{\boldsymbol{w}}_1, \dots, \tilde{\boldsymbol{w}}_k$ are randomly generated in a bounded subset W of $\mathbb{R}^n$ (see Section III-B), the probability of choosing a sequence $\tilde{\boldsymbol{w}}_{\to k}$ in $S(\tilde{\boldsymbol{w}}^*_{\to k})$ tends to 1 as the number of iterations increases. In view of the previous arguments, this concludes the proof. ■

VI. Extension: Achieving Smoothness

The randomized planner presented so far produces paths in $\mathcal{C}$ that are only piecewise-differentiable with respect to s. This is due to the piecewise-constant choice of the residual input vector $\tilde{\boldsymbol{w}}$ in (5), which implies that the geometric velocity $\boldsymbol{q}'$ will be discontinuous in correspondence to the discrete parameter values $s_1, \ldots, s_N$.

The isolated discontinuities of $\boldsymbol{q}'$ may not represent necessarily a problem in practice. One way to handle them would be to associate to the solution path a timing law $s = s(t)$ that slows down sufficiently at $s_1, \ldots, s_N$ to allow the robot actuators to track accurately the corresponding trajectory. Another possibility would be to perform a controlled *smoothing* of the solution path at $s_1, \ldots, s_N$, to recover differentiability while keeping the task error within reasonable bounds.

However, it is also possible to modify the proposed planner and specifically the motion generation schemes at its core, to directly produce continuously differentiable paths, which we simply call *smooth* in the following. The modifications concern both the forward/backward tree extension and the loop closure, and are discussed in detail below.

A. Smooth Tree Extension

The motion generation scheme introduced in Section III-B for tree extension can be easily modified to produce smooth paths in the configuration space. To this end, we still use (5) but with a piecewise-linear (rather than piecewise-constant) profile of the residual input vector $\tilde{\boldsymbol{w}}$ over s. This is obtained as follows.

Extension of the forward tree $T_{\rm fw}$ starts from $(\boldsymbol{q}_{\rm ini}, s_1)$ plugging in (5) a value of $\tilde{\boldsymbol{w}}$, which is chosen randomly in W and kept constant for $s \in [s_1, s_2]$. The same value of residual input, with opposite sign, is used to extend the backward tree $T_{\rm bw}$ from $(\boldsymbol{q}_{\rm ini}, s_N)$ over $s \in [s_N, s_{N-1}]$. Given the smoothness of $\boldsymbol{y}_d$, continuity of $\tilde{\boldsymbol{w}}$ will result in continuity of $\tilde{\boldsymbol{v}}$, and, in turn, this guarantees that paths in $T_{\rm fw}$ and $T_{\rm bw}$ join smoothly at $\boldsymbol{q}_{\rm ini}$.

Further extension of $T_{\rm fw}$ and $T_{\rm bw}$ is made in such a way that the same value of $\tilde{\boldsymbol{w}}$ is used when entering and exiting a node. For example, assume that we are extending $T_{\rm fw}$ from $\boldsymbol{q}_{\rm near}$ on $\mathcal{L}_i$ (the leaf of $\mathcal{C}_{\rm task}$ associated to $s = s_i$) towards $\mathcal{L}_{i+1}$. Denote by $\tilde{\boldsymbol{w}}_{\rm near}$ the value of $\tilde{\boldsymbol{w}}$ associated with $\boldsymbol{q}_{\rm near}$, i.e., the final value of $\tilde{\boldsymbol{w}}$ along the subpath leading to $\boldsymbol{q}_{\rm near}$ (this value is stored whenever a node is created). We choose a random value in W for $\tilde{\boldsymbol{w}}(s_{i+1})$, and then, we set

$$\tilde{\boldsymbol{w}}(s) = \tilde{\boldsymbol{w}}_{\rm near} + \psi\,(s - s_i), \qquad s \in [s_i, s_{i+1}],$$

where $\psi = (\tilde{\boldsymbol{w}}(s_{i+1}) - \tilde{\boldsymbol{w}}_{\rm near})/(s_{i+1} - s_i)$. This will produce a linear transition of $\tilde{\boldsymbol{w}}$ from $\tilde{\boldsymbol{w}}_{\rm near}$ to $\tilde{\boldsymbol{w}}(s_{i+1})$.

As before, continuity of $\tilde{\boldsymbol{w}}$ implies that all subpaths in $T_{\rm fw}$ and $T_{\rm bw}$ are smooth at $s_2, \ldots, s_{N-1}$; moreover, as explained before, they also join smoothly at $\boldsymbol{q}_{\rm ini}$.

The above mechanism still guarantees that all residual inputs $\tilde{\boldsymbol{w}}$ are contained in the bounded subset W, and therefore the previous proof of probabilistic completeness holds also when this modified scheme is used for tree extension.

B. Smooth Loop Closure

To achieve smooth loop closure, we must join $\boldsymbol{q}_{\rm fw} \in \mathcal{L}_k$ and $\boldsymbol{q}_{\rm bw} \in \mathcal{L}_{k+1}$, respectively, nodes of $\mathcal{T}_{\rm fw}$ and $\mathcal{T}_{\rm bw}$ with a subpath that respects the boundary geometric velocities $(\boldsymbol{q})'_{\rm fw}$ and $(\boldsymbol{q})'_{\rm bw}$. Clearly, this path must also realize the portion of task between $\boldsymbol{y}_{d,k}$ and $\boldsymbol{y}_{d,k+1}$, and be collision-free (see Fig. 1).

The idea is replace the pure feedback scheme (7), (8) (or (7)–(11)) with a hybrid mechanism, in which motion for the $(n-m)$-dimensional vector $\boldsymbol{q}_r$ is directly generated as a subpath that joins $\boldsymbol{q}_{r,\rm fw}$ and $\boldsymbol{q}_{r,\rm bw}$ with the appropriate boundary conditions. To this end, we use an interpolation scheme. A straightforward choice is to define q_r^i, the ith component of vector $\boldsymbol{q}_r$, as a cubic polynomial

$$q_r^i(s) = a_i(s - s_k)^3 + b_i(s - s_k)^2 + c_i(s - s_k) + d_i, \qquad s \in [s_k, s_{k+1}],$$

with scalars $a_i, \ldots, d_i$ computed so as to impose the boundary conditions

$$\begin{aligned} q_r^i(s_k) &= q_{r,\rm fw}^i \\ (q_r^i)'(s_k) &= (q_r^i)'_{\rm fw} \\ q_r^i(s_{k+1}) &= q_{r,\rm bw}^i \\ (q_r^i)'(s_{k+1}) &= (q_r^i)'_{\rm bw}. \end{aligned}$$

Once $\boldsymbol{q}_r$ has been computed, the analytic expression of the geometric velocity $\boldsymbol{q}'_r$ over $[s_k, s_{k+1}]$ is easily derived.

Motion for the $\boldsymbol{q}_b$ coordinates is then generated by integrating the lower part of (6) from $(\boldsymbol{q}_{b,\rm fw}, s_k)$ under the same feedback control as in the lower part of (7):

$$\boldsymbol{q}'_b = -\boldsymbol{J}_{t,b}^{-1}\boldsymbol{J}_{t,r}\boldsymbol{q}'_r + \boldsymbol{J}_{t,b}^{-1}(\boldsymbol{y}'_d + k_t\boldsymbol{e}_t).$$

Therefore, smoothness of $\boldsymbol{y}_d$ (by assumption) and of $\boldsymbol{q}_r$ (by construction) will guarantee that also $\boldsymbol{q}_b$ is smooth.

The smooth version of the actual loop closure procedure is then obtained exactly as in Section IV-C, the only difference being that the motion to be verified is generated as explained above. It is important to understand that Proposition 1 is also valid for the smooth loop closure procedure. Therefore, the latter will work under the same assumptions in which the standard procedure does.

VII. Planning Results

We implemented the R-TCMP planner as a C++ plugin for Kite, a cross-platform motion planning software produced by KineoCAM (now Siemens PLM), on a 64-bit Intel Core i7-2600 CPU running at 3.4 GHz under a 32-bit operating system.

We will illustrate the performance of the planner by presenting numerical results in three different scenarios. Two of these involve free-flying robots (a fixed-base manipulator and an omnidirectional mobile manipulator), while a nonholonomic mobile manipulator is considered in the third. All the results are also shown in the accompanying video clip.

Table I reports the values used for the planner parameters in the three scenarios. In particular, we use the following symbols.

TABLE I
R-TCMP PLANNER PARAMETERS

Scenario (path length)	N	k_t	k_r (automatic)	δs	α
1 (0.99 m)	11	100	not used	0.002	150%
2 (7.18 m)	21	100	12.27	0.002	600%
3 (10 m)	11	100	12.92	0.002	300%

TABLE II
R-TCMP PERFORMANCE DATA (AVERAGED)

Scenario	Execution time	Nodes in T_{fw}	Nodes in T_{bw}	Collision tests	Mean task error
1	19 s	20	19	9587	0.06 mm
2	133 s	48	94	17710	0.32 mm
3	89 s	90	176	109779	0.57 mm

1) N is the number of equispaced sample points chosen on the assigned task path (see the beginning of Section III).
2) k_t is the task error gain in both (5) and (7).
3) k_r, used only in Scenarios 2 and 3, is the error gain given by (12) for loop closure via (11), with $\eta = 1/2$.
4) δs is the stepsize used for reconstructing joint motions via Euler integration.
5) α is the maximum admissible norm of the second term in (5) (the null-space velocity) as a percentage of the norm of the first term. This bound induces a bound on the norm of the random residual input $\tilde{\boldsymbol{w}}$, which will then take values in a bounded subset W of $\mathbb{R}^n$ as postulated in Section III-B and required by the proof of probabilistic completeness (see the previous section).

Note that setting $\alpha = 0$ corresponds to the pure pseudoinverse solution, i.e., using a zero null-space velocity in (5). Hence, the larger α, the larger the amount of exploration of the solution space that the planner is allowed to perform. One may, therefore, look at choosing α as the classical problem of finding a balance between exploitation and exploration. While it is, in principle, possible to choose α adaptively within the planner (e.g., using reinforcement learning), we have simply used a constant α throughout each run of the planner.

Table II collects the most significant performance data in the three scenarios. Since our planner is randomized, these data have been averaged over ten runs.

A. Scenario 1: A Manipulator Drawing a Circle

In the first scenario (see Fig. 4), a 7-dof KUKA LWR-IV manipulator must draw an ellipse on a whiteboard using a felt tip marker. The configuration vector is $\boldsymbol{q} = (\theta_1, \ldots, \theta_7)$, where θ_i is the ith joint variable. The task is assigned as an elliptic path, parameterized by $s \in [s_{\mathrm{ini}} = 0, s_{\mathrm{fin}} = 1]$, in the 3-dimensional task space (tip positioning). The degree of redundancy is then $7 - 3 = 4$, but the planning problem is made difficult by the location of the robot, which is mounted on a platform placed behind the wall and must pass through a small round opening to reach the whiteboard. To allow the robot to execute the drawing task, simple contacts between the marker tip and the whiteboard are obviously allowed.

For this scenario, we have used the modified planner of Section VI to obtain smooth paths in configuration space. The planner parameters are chosen as in row 1 in Table I. Note that no value is given for k_r because loop closure is realized with the hybrid mechanism described in Section VI-B. Distances in $\mathcal{C}$ are computed using the metric induced by the L_1 norm

$$d(\boldsymbol{q}_A, \boldsymbol{q}_B) = \sum_{i=1}^{7} \min \{|\theta_{i,A} - \theta_{i,B}|, 2\pi - |\theta_{i,A} - \theta_{i,B}|\}.$$

The solution shown in Fig. 4 is clearly repeatable, because the first and last robot configurations are identical. This is confirmed by Fig. 5, which shows that all joint displacements $\Delta\theta_i = \theta_i - \theta_{i,\mathrm{ini}}$ return to zero at the end of the motion. Note also the expected continuously differentiable profile; in particular, the tangents (i.e., the geometric velocities) at $s = 0$ and $s = 1$ are the same, indicating that the robot transition from one cycle to the next will be smooth. Loop closure takes place for $s \in [0.5, 0.6]$. The actual motion of the robot is better appreciated in the accompanying video.

On the average, the planner took 19 s to find this kind of solution (see row 1 in Table II). Note the very low value (0.06 mm) of the mean task error, which is computed over the whole joint motion; this proves that we have indeed achieved exact tracking in practice. However, it should be considered that we deal with motion planning and do not address issues related to sensor-based real-time execution of the generated motions. Therefore, the above precision simply indicates that arbitrary accuracy can be achieved at the planning level.

The accompanying video contains also a clip showing an experiment with an actual KUKA LWR-IV manipulator performing a similar motion plan. In addition to proving the dynamic feasibility of the planned trajectory, the experiment also highlights the role of repeatability for safe coexistence between the robot and a human.

B. Scenario 2: A Mobile Manipulator Moving Its End-Effector on an Elliptical Path

The robot considered in the second scenario (see Fig. 6) is a KUKA youBot, a mobile manipulator consisting of an omnidirectional base carrying a 5-dof arm. The configuration vector is $\boldsymbol{q} = (x, y, \theta_b, \theta_1, \ldots, \theta_5)$, where x and y are the Cartesian coordinates of a representative point of the base, θ_b is the orientation of the base, and θ_i is the ith joint variable of the arm. The assigned task path is a horizontal ellipse, parameterized by $s \in [s_{\mathrm{ini}} = 0, s_{\mathrm{fin}} = 1]$, that the robot must follow with its end-effector (a two-finger gripper). Several cone-shaped obstacles are placed on the ground. We must, therefore, plan for a free-flying robot ($r = 0$) with $n = 8$ and $m = 3$, and the degree of redundancy is $8 - 3 = 5$.

For this scenario, as for the next, we used the original version of the planner described in Sections III–IV. The parameters for the planner are chosen as in row 2 in Table I. In view of the total length of the path (7.18 m), the number of samples from the task path is increased to $N = 21$. A larger α is also used to

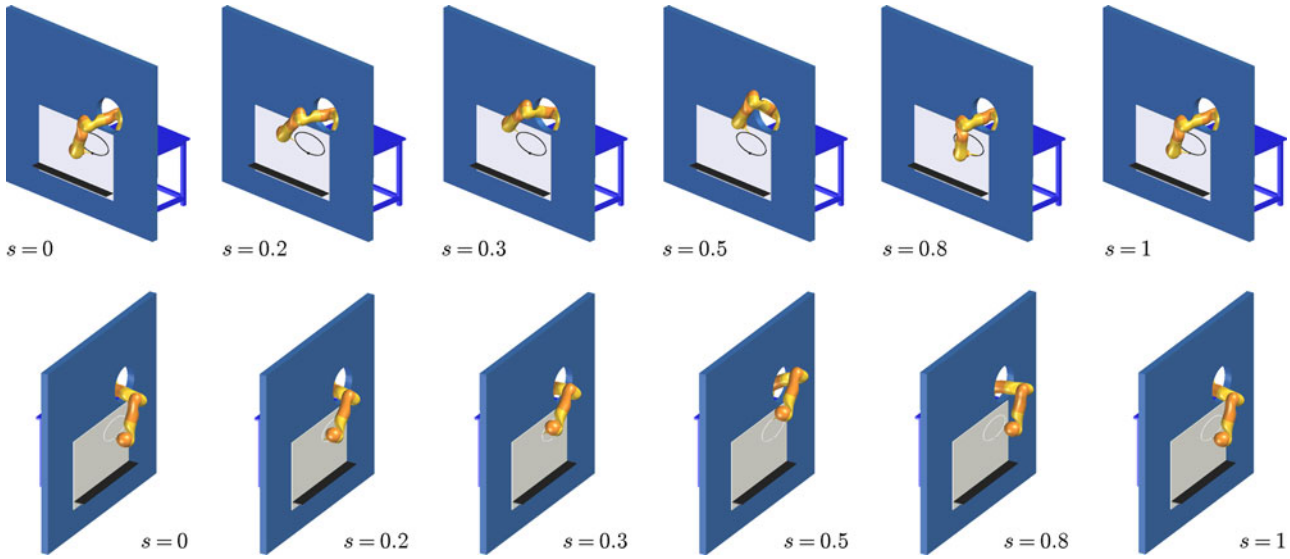

Fig. 4. Scenario 1: A KUKA LWR-IV manipulator drawing an ellipse on a whiteboard. Six snapshots from a solution are shown, taken from two different points of view. The first ($s = 0$) and last ($s = 1$) snapshots confirm that repeatability has been achieved.

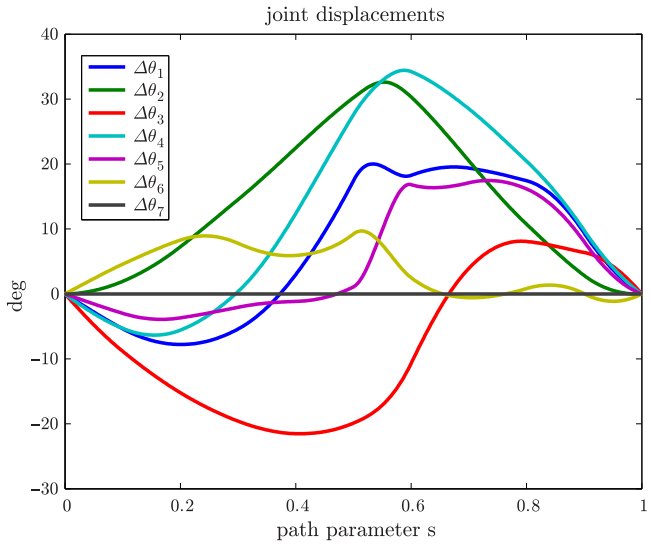

Fig. 5. Scenario 1: joint displacements with respect to $\boldsymbol{q}_{\mathrm{ini}}$.

give the planner more leeway in the random exploration of the space of possible solutions.

The following metric is used in $\mathcal{C}$:

$$d(\boldsymbol{q}_A, \boldsymbol{q}_B) = \lambda\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} + \min\{|\theta_{b,A} - \theta_{b,B}|, 2\pi - |\theta_{b,A} - \theta_{b,B}|\} + \sum_{i=1}^{5} \min\{|\theta_{i,A} - \theta_{i,B}|, 2\pi - |\theta_{i,A} - \theta_{i,B}|\}.$$

The role of λ is to properly weigh the norm of the Cartesian versus angular coordinates.

A typical result is shown in Figs. 6 and 7: again, repeatability has been achieved. Loop closure takes place for $s \in [0.5, 0.55]$. The actual motion of the robot is fully visible in the accompanying video. Row 2 of Table II reports a predictable increase in the time needed to find solutions. The mean task error, on the other hand, is still very low.

C. Scenario 3: A Nonholonomic Mobile Manipulator Performing a "Back and Forth" Task

In the third scenario (see Fig. 8), we consider a nonholonomic mobile manipulator consisting of a differential-drive base carrying a KUKA LWR-IV. The robot is assigned a task of possible interest in an industrial context: it must travel back and forth between two end-effector placements,[9] the first in front of a window opening on a human workstation and the second over a repository of tools that are needed by the operator. Since the line of sight between the placements is collision-free, the path has been simply assigned as a line segment between them, to be followed first in the forward direction ($s \in [s_{\mathrm{ini}} = 0, 0.5]$) and then in the backward direction ($s \in [0.5, s_{\mathrm{fin}} = 1]$). The generation of a repeatable collision-free path in configuration space is left[10] to the planner.

The configuration vector is $\boldsymbol{q} = (x, y, \theta_b, \theta_1, \ldots, \theta_7)$, where x and y are the Cartesian coordinates of a representative point of the base, θ_b is the orientation of the base, and θ_i is the ith joint variable of the arm. The configuration space has, therefore, dimension $n = 3 + 7 = 10$, but the robot is not free-flying due to the nonholonomic constraint acting on the base ($r = 1$), which reduces the number of effective dofs ($p = 10 - 1 = 9$).

[9]One can imagine that such placements have been computed to optimize certain optimality criteria, related, respectively, to the interaction with the human and the ease of tool grasping.

[10]The cyclic task path of this scenario is made by two identical parts (forward and backward path). In this very special case, a trivial repeatable solution can be obtained by planning for the forward part and simply reversing joint motions in the backward part. Our planner, however, was able to compute nontrivial solutions.

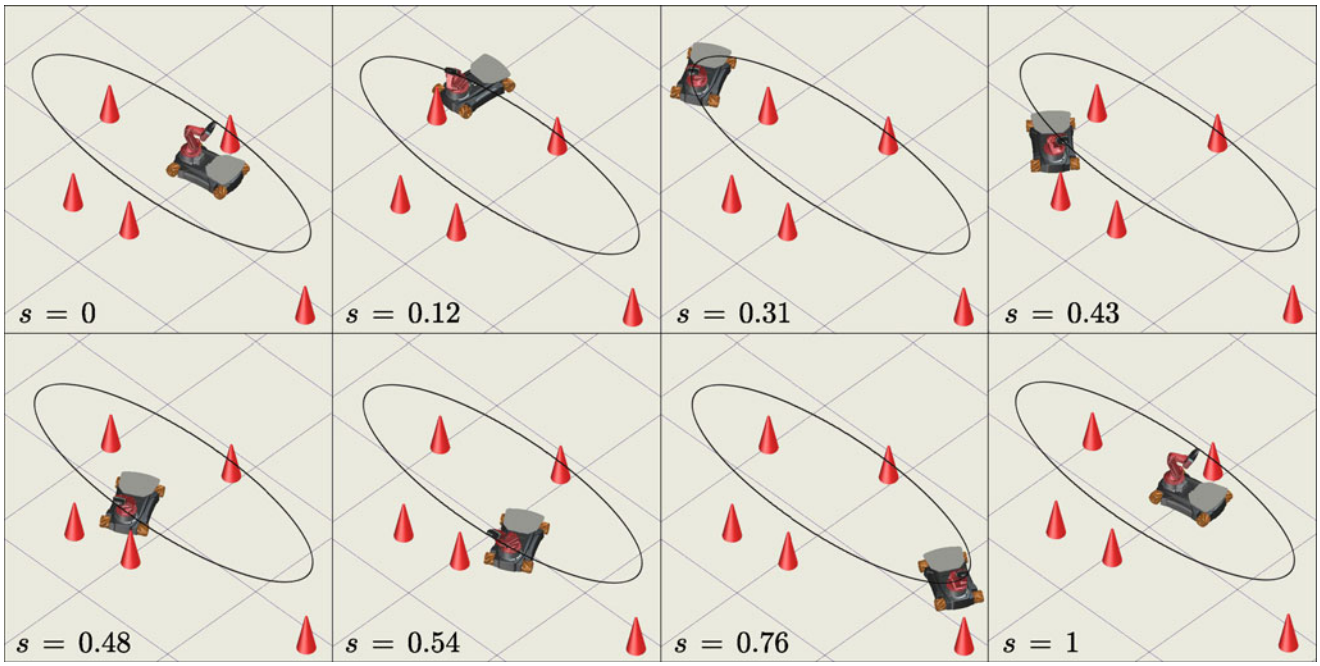

Fig. 6. Scenario 2: A KUKA youBot moving its end-effector on an elliptical path. Eight snapshots from a solution are shown. The first ($s = 0$) and last ($s = 1$) snapshot confirm that repeatability has been achieved.

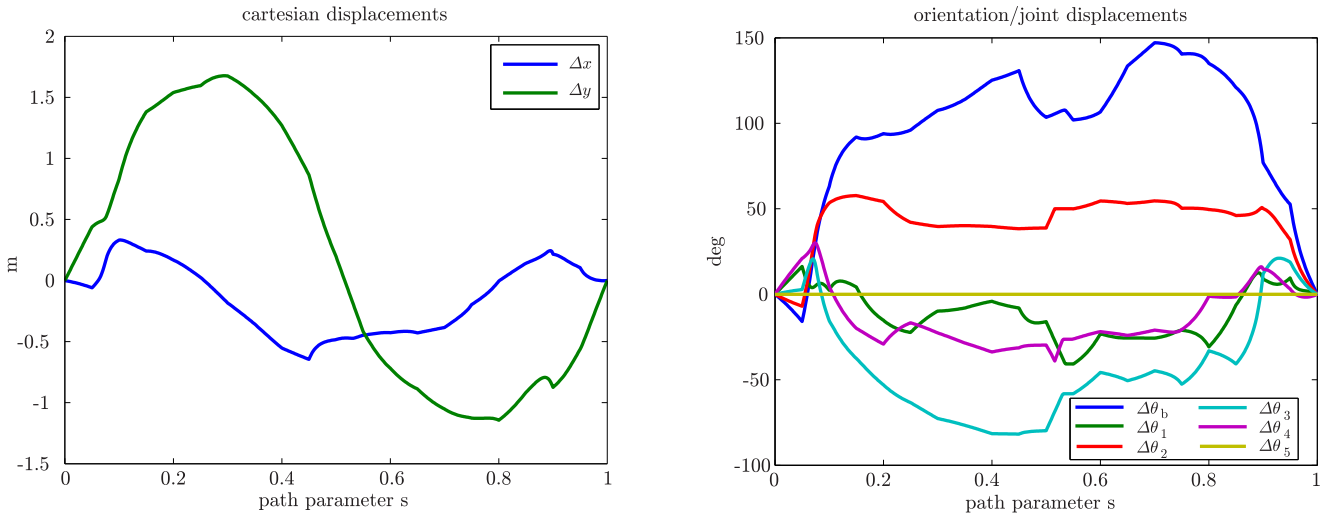

Fig. 7. Scenario 2: Cartesian (left) and orientation/joint displacements (right) with respect to $\boldsymbol{q}_{\mathrm{ini}}$.

In particular, we have $n_c = 3$ constrained coordinates (those of the base) and $n_u = 7$ unconstrained coordinates (those of the manipulator). Since the dimension of the task is $m = 3$, the degree of (kinematic) redundancy in this case is $9 - 3 = 6$.

The planner parameters are chosen as in row 3 in Table I. The metric in $\mathcal{C}$ is defined similarly to the second scenario.

Computing a repeatable solution such as the one shown in Figs. 8 and 9, and fully visible in the accompanying video, takes an average of 89 s.

For this solution, it is interesting to look at the loop closure automatically produced by the planner for $s \in [0.4, 0.5]$ using the following partition of $\boldsymbol{q} = (\boldsymbol{q}_r, \boldsymbol{q}_b)$:

$$\begin{aligned} \boldsymbol{q}_r &= (\theta_b, \theta_1, \theta_2, \theta_3, \theta_4, \theta_6, \theta_7) \\ \boldsymbol{q}_b &= (x, y, \theta_5). \end{aligned}$$

Since all the variables in $\boldsymbol{q}_r$ are governed by simple integrator dynamics, they can be directly steered using the finite-time control law (11). No adaptation was, therefore, necessary for this specific loop closure.

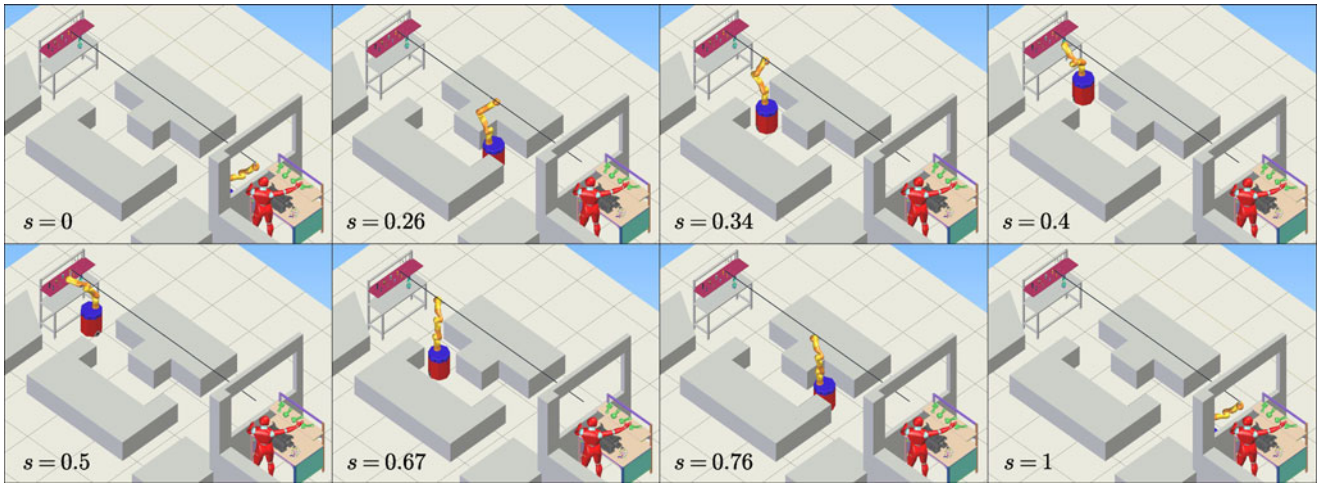

Fig. 8. Scenario 3: A nonholonomic mobile manipulator performing a "back and forth" task. Eight snapshots from a solution are shown. The first ($s = 0$) and last ($s = 1$) snapshot confirm that repeatability has been achieved.

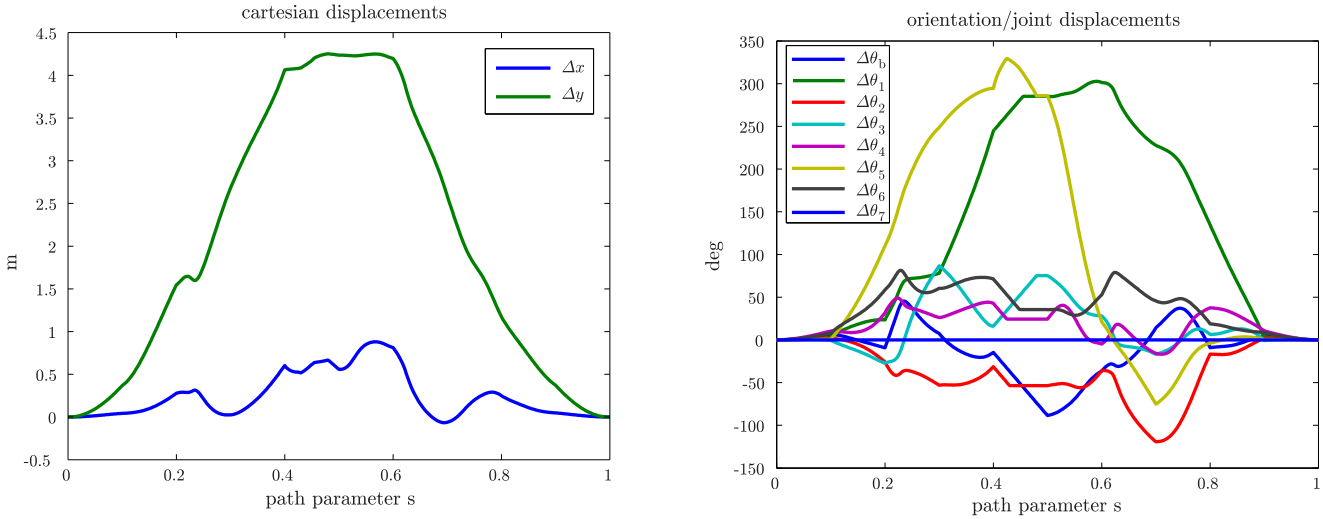

Fig. 9. Scenario 3: Cartesian (left) and orientation/joint displacements (right) with respect to $\boldsymbol{q}_{\mathrm{ini}}$.

VIII. Conclusion

For redundant robotic systems subject to cyclic task constraints, we have presented a control-based approach for planning repeatable, collision-free paths in configuration space. We argue that our contribution solves an open problem in the literature. In fact, on the one hand, kinematic control schemes are generally nonrepeatable, and when they are, they cannot claim guaranteed obstacle avoidance. On the other hand, existing task-constrained motion planners are not designed for mapping cyclic task paths to repeatable motions in the robot configuration space.

Our randomized R-TCMP planner relies on bidirectional search and loop closure in the task-constrained configuration space. As a consequence, it produces closed paths, on which continued satisfaction of the task constraint is guaranteed. Its probabilistic completeness has been proven explicitly, and a modified version of the planner has been designed for producing configuration-space paths that are also smooth. Planning results on various scenarios involving both free-flying and nonholonomic robots have been presented to show the effectiveness of the proposed method.

In this paper, we have assumed that a task path or trajectory is assigned, and that the obstacles are stationary. An interesting extension would be to consider the case of obstacles moving along known trajectories. If the task is a trajectory ($s = t$), then the proposed method can be applied as is. If the task is assigned as a path, however, the planner should also generate a time history $s = s(t)$ to take advantage of the possibility of slowing down/speeding up along the path. To this end, one may adopt the framework in [33].

Other future work on this topic may address several points, including the following:

1) inclusion of torque constraints, moving the proposed approach to the acceleration level, as in [34];
2) application of this approach to the synthesis of repeatable gaits for humanoids.

REFERENCES

[1] S. Chiaverini, G. Oriolo, and I. Walker, "Kinematically redundant manipulators," in *Handbook of Robotics*, O. Khatib and B. Siciliano, Eds. Berlin, Germany: Springer, 2009, ch. 11, pp. 245–268.

[2] A. Dragan, K. C. Lee, and S. Srinivasa, "Legibility and predictability of robot motion," in *Proc. 8th ACM/IEEE Int. Conf. Human-Robot Interact.*, Tokyo, Japan, 2013, pp. 301–308.

[3] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *J. Intell. Robot. Syst.*, vol. 3, pp. 201–212, 1990.

[4] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, no. 12, pp. 868–871, Dec. 1977.

[5] O. Egeland, "Task-space tracking with redundant manipulators," *IEEE J. Robot. Autom.*, vol. RA-3, no. 5, pp. 471–475, Oct. 1987.

[6] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

[7] C. A. Klein and C. H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 2, pp. 245–250, Mar./Apr. 1983.

[8] T. Shamir and Y. Yomdin, "Repeatability of redundant manipulators: Mathematical solution of the problem," *IEEE Trans. Autom. Control*, vol. 33, no. 11, pp. 1004–1009, Nov. 1988.

[9] R. Schaufler, C. Fedrowitz, and R. Kammuller, "A simplified criterion for repeatability and its application in constraint path planning problems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Takamatsu, Japan, 2000, pp. 2345–2350.

[10] A. De Luca, L. Lanari, and G. Oriolo, "Control of redundant robots on cyclic trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, Nice, France, 1992, pp. 500–506.

[11] R. G. Roberts and A. A. Maciejewski, "Repeatable generalized inverse control strategies for kinematically redundant manipulators," *IEEE Trans. Autom. Control*, vol. 38, no. 5, pp. 689–699, May 1993.

[12] R. Mukherjee, "Design of holonomic loops for repeatability in redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, Nagoya, Japan, 1995, pp. 2785–2790.

[13] Y. Michellod, P. Mullhaupt, and D. Gillet, "On achieving periodic joint motion for redundant robots," in *Proc. IFAC World Congr.*, Seoul, South Korea, 2008, pp. 4355–4360.

[14] A. M. Zanchettin and P. Rocco, "A general user-oriented framework for holonomic redundancy resolution in robotic manipulators using task augmentation," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 514–521, Apr. 2012.

[15] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, St. Louis, MO, USA, 1985, pp. 722–728.

[16] P. H. Chang, "A closed-form solution for inverse kinematics of robot manipulators with redundancy," *IEEE Trans. Robot. Autom.*, vol. RA-3, no. 5, pp. 393–403, Oct. 1987.

[17] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, 2002, pp. 1657–1662.

[18] G. Oriolo and C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 2166–2172.

[19] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, CA, USA, 2007, pp. 3074–3081.

[20] G. Oriolo and M. Vendittelli, "A control-based approach to task-constrained motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, St. Louis, MO, USA, 2009, pp. 297–302.

[21] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. J. Robot. Res.*, vol. 30, no. 12, pp. 1435–1460, 2011.

[22] C. Suh, B. Kim, and F. Park, "The tangent bundle RRT algorithms for constrained motion planning," in *Proc. 13th World Congr. Mech. Mach. Sci.*, Guanajuato, Mexico, 2011, pp. 1–5.

[23] L. Jaillet and J.-M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 105–117, Feb. 2013.

[24] S. Lengagne, N. Ramdani, and P. Fraisse, "A new method for generating safe motions for humanoid robots," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots*, Daejeon, South Korea, 2008, pp. 105–110.

[25] M. Cefalo, G. Oriolo, and M. Vendittelli, "Planning safe cyclic motions under repetitive task constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 3807–3812.

[26] A. De Luca, G. Oriolo, and P. Robuffo Giordano, "Image-based visual servoing schemes for nonholonomic mobile manipulators," *Robotica*, vol. 25, no. 2, pp. 129–145, 2007.

[27] J. W. Burdick, "On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds," in *Proc. IEEE Int. Conf. Robot. Autom.*, Scottsdale, AZ, USA, 1989, pp. 264–270.

[28] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[29] G. Oriolo, "Stabilization of self-motions in redundant robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Diego, CA, USA, 1994, pp. 704–710.

[30] P. Wenger, "Classification of 3R positioning manipulators," *J. Mech. Des.*, vol. 120, pp. 327–332, 1998.

[31] S. Bhat and D. Bernstein, "Finite-time stability for continuous autonomous systems," *SIAM J. Control Optim.*, vol. 38, no. 3, pp. 751–766, 2000.

[32] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, U.K.: Springer, 2009.

[33] M. Cefalo, G. Oriolo, and M. Vendittelli, "Task-constrained motion planning with moving obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, 2013, pp. 5758–5763.

[34] M. Cefalo and G. Oriolo, "Dynamically feasible task-constrained motion planning with moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, 2014, pp. 2045–2050.

Giuseppe Oriolo (S'89–M'92–SM'02–F'16) received the Ph.D. degree in control engineering from Sapienza University of Rome, Italy, in 1992.

He is currently with the Department of Computer, Control and Management Engineering (DIAG), Sapienza University of Rome, as a Full Professor of automatic control and robotics and the Coordinator of the DIAG Robotics Lab. His research interests are in the general area of planning and control of robotic systems.

Prof. Oriolo has been an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION from 2001 to 2005, and an Editor of the IEEE TRANSACTIONS ON ROBOTICS from 2009 to 2013.

Massimo Cefalo received the Ph.D. degree in control engineering from Sapienza University of Rome, Italy, in 2006.

He currently holds a Postdoctoral position with the Department of Computer, Control and Management Engineering (DIAG), Sapienza University of Rome. His research interests include robot motion and path planning, real-time algorithms, control of underactuated mechanical systems, and e-learning.

Marilena Vendittelli received the Ph.D. degree in systems engineering from Sapienza University of Rome, Italy, in 1997.

She is an Associate Professor at Sapienza University of Rome, Department of Information Engineering, Electronics and Telecommunications, and a member of the Robotics Lab, at the Department of Computer, Control and Management Engineering (DIAG). She teaches Automatic Control and Medical Robotics. Her main research interests are in robot motion planning and control, with an emphasis on nonholonomic and redundant systems. She held a Postdoctoral position at LAAS-CNRS, Toulouse, France, from 1997 to 1998, funded by Marie Curie Research Training Grants, and she joined DIAG in 2001 as an Assistant Professor.

Dr. Vendittelli has been an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS from 2010 to 2013.