The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

A control-based approach to task-constrained motion planning

Giuseppe Oriolo Marilena Vendittelli

Abstract— **We consider the problem of planning collision-free motions for general (i.e., possibly nonholonomic) redundant robots subject to task space constraints. Previous approaches to the solution are based on the idea of sampling and inverting the task constraint to build a roadmap of task-constrained configurations which are then connected by simple local paths; hence, task tracking is not enforced during the motion between samples. Here, we present a control-based randomized approach relying on a motion generation scheme that guarantees continued satisfaction of such constraint. The resulting planner allows to achieve accurate execution of the desired task without increasing the size of the roadmap. Numerical results on a fixed-base manipulator and a free-fying mobile manipulator are presented to illustrate the performance improvement obtained with the proposed technique.**

I. Introduction

Task space constraints invariably arise in the practical operation of robotic systems, both in service and industrial applications; examples include opening a door, transporting an object, cooperating with other robots, executing a given end-effector trajectory for drawing, cutting or welding, tracking a visual target. Kinematically redundant robotic systems, such as humanoids and mobile manipulators, possess the dexterity for accomplishing these tasks while pursuing additional objectives, among which the most important is obstacle avoidance. A motion planner should be able to generate robot motions that satisfy the task space constraints while guaranteeing that the robot body does not collide with parts of itself (self-collision) or with workspace obstacles. In the following, this problem is referred to as Task-Constrained Motion Planning (TCMP).

Researchers initially attacked the TCMP problem as a special case of redundancy resolution using either local or global optimization techniques; see [1], [2], [3] for general reviews of optimization-based redundancy resolution. Both these approaches to TCMP proved to be unpractical for realistic motion planning, the first due to the presence of local minima and the second because it leads to a nonlinear TPBVP whose solution can only be seeked (without guarantee of success) via numerical techniques.

To overcome these limitations, in [4] we applied the principles of randomized planning to develop a solution for the TCMP problem in the special case of fixed-base redundant manipulators subject to end-effector path constraints. In [5], this approach was extended to the case of nonholonomic mobile manipulators. Along the same lines are the techniques presented in [6], where a unified representation of end-effector constraints is also proposed. A related problem is motion planning for closed kinematic chains, in which the closure condition may be considered as a constraint; randomized techniques for this problem have been presented in [7], [8], [9].

All the above planning techniques are based on the idea of randomly sampling and inverting the constraint to build a roadmap consisting of admissible configurations; these are then connected by simple local paths (typically, a linear local planner is used). Hence, task tracking is not enforced during the motion between samples. A higher sampling rate in the task space can reduce the tracking error at the price of an increased size of the roadmap and, as a consequence, of the time needed to solve the planning problem.

Our solution to the TCMP problem relies on the principle of *control-based* motion planning [10], a paradigm where configuration samples are generated using a differential model of the robot (called *motion generation scheme* in the following). In addition to the system dynamics, often represented as simple chains of integrators, this model can incorporate velocity/acceleration bounds (*kinodynamic* motion planning [11], [12]) as well as non-integrable kinematic constraints (*nonholonomic* motion planning [13], [14]). In this paper, we extend the motion generation scheme so as to guarantee continued satisfaction of the constraint; this leads to an RRT-based planner that achieves accurate execution of the task without increasing the size of the roadmap.

The paper is organized as follows. Sect. II presents some background material on task kinematics and redundancy concepts. The TCMP problem is formulated in Sect. III, and our motion generation scheme is introduced in Sect. IV. The control-based planner for solving the TCMP problem is described in Sect. V. Numerical results are presented in Sect. VI, while extensions and future work are briefly discussed in the conclusions.

II. Task Kinematics and Redundancy

Consider a robotic system whose configuration $\boldsymbol{q}$ takes value in an n_q-dimensional configuration space $\mathcal{Q}$. For the sake of generality, partition $\boldsymbol{q}$ as $(\boldsymbol{q}_a^T \quad \boldsymbol{q}_b^T)^T$, where $\boldsymbol{q}_a$ is n_a-dimensional and $\boldsymbol{q}_b$ is n_b-dimensional, and assume that $\boldsymbol{q}_a$ is subject to n_c nonholonomic constraints of the Pfaffian form

$$\boldsymbol{A}(\boldsymbol{q}_a)\dot{\boldsymbol{q}}_a = 0. \tag{1}$$

According to the partition, it is $\mathcal{Q} = \mathcal{Q}_a \times \mathcal{Q}_b$. The above description encompasses most robotic systems, including

G. Oriolo and M. Vendittelli are with the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Ariosto 25, 00185 Roma, Italy. E-mail: {oriolo,venditt}@dis.uniroma1.it

fixed-base manipulators, wheeled or legged mobile robots, and mobile manipulators.

The velocity constraint (1) has a *geometric* counterpart [15], in that any admissible configuration space path $\boldsymbol{q}(s)$, where s is a path parameter, must be such that

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

with the notation $(\ )' = d(\ )/ds$. For motion planning[1] purposes, it is therefore appropriate to use the geometric form of the system kinematic model, which specifies the admissible tangent vectors to the path as

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

where $\boldsymbol{G}(\boldsymbol{q}_a)$ is an $n_a \times (n_a - n_c)$ matrix whose columns are a basis for the null space of $\boldsymbol{A}(\boldsymbol{q}_a)$, $\tilde{\boldsymbol{v}}_a$ is $(n_a - n_c)$-dimensional, and $\tilde{\boldsymbol{v}}_b$ is n_b-dimensional. This model entails the simple fact that the geometric motion of the $\boldsymbol{q}_a$ coordinates in $\mathcal{Q}_a$ is locally constrained by (2), whereas the $\boldsymbol{q}_b$ coordinates can move in arbitrary directions of $\mathcal{Q}_b$. The tilde over the $\boldsymbol{v}$'s is a reminder that these are *geometric* inputs, rather than velocity inputs. The driftless system (3) is controllable in view of the nonholonomy of the constraints (2).

Consider now a set of task coordinates $\boldsymbol{t}$ taking values in an n_t-dimensional space $\mathcal{T}$. Typical robotic tasks concern manipulation (end-effector position and/or orientation) or perception (sensor pose or even features in the sensing space, as in visual servoing). The task coordinates are related to the configuration coordinates by the kinematic map

$$\boldsymbol{t} = \boldsymbol{f}(\boldsymbol{q}). \tag{4}$$

At the differential level, we have

$$\boldsymbol{t}' = \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{q})\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{f},a}(\boldsymbol{q}) & \boldsymbol{J}_{\boldsymbol{f},b}(\boldsymbol{q}) \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_a' \\ \boldsymbol{q}_b' \end{pmatrix}$$

where $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{q}) = d\boldsymbol{f}/d\boldsymbol{q}$, and using (3)

$$\boldsymbol{t}' = \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{f},a}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q}_a) & \boldsymbol{J}_{\boldsymbol{f},b}(\boldsymbol{q}) \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{v}}_a \\ \tilde{\boldsymbol{v}}_b \end{pmatrix} = \boldsymbol{J}(\boldsymbol{q})\tilde{\boldsymbol{v}}. \tag{5}$$

Let $n = n_q - n_c$ be the number of degrees of freedom of the robot. The $n_t \times n$ matrix $\boldsymbol{J}(\boldsymbol{q})$, simply called *task Jacobian*[2] in the following, maps the available geometric inputs to the admissible task-space tangent vectors [16].

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

- *static* redundancy occurs when $n_q > n_t$ (the number of configuration coordinates exceeds the task dimension);
- *kinematic* redundancy occurs when $n > n_t$ (the number of degrees of freedom exceeds the task dimension).

These two concepts collapse in the absence of nonholonomic constraints (e.g., for fixed-base manipulators), that implies $n = n_q$. In general, kinematic redundancy implies static redundancy, whereas the converse is not true; for example, a mobile manipulator consisting of a unicycle with a rigidly attached gripper is statically but not kinematically redundant for planar positioning tasks.

If the robot is statically redundant with respect to the task, the inverse image $\bar{\boldsymbol{q}} = \boldsymbol{f}^{-1}(\bar{\boldsymbol{t}})$ of a point $\bar{\boldsymbol{t}}$ in the task space may be either *(a)* an $(n_q - n_t)$-dimensional subset of $\mathcal{C}$, consisting of one or more disjoint manifolds, or *(b)* a finite number of configurations [17]. Task points of the first group include *regular* points and *coregular* points (also called avoidable singularities), whereas the second group consists of *singular* points (also called unavoidable singularities).

III. The TCMP Problem

Consider a robotic system in the form (3), and suppose that it is kinematically redundant for the task of interest, related to the configuration variables by eq. (4). Assume that a desired path is assigned for the task variables $\boldsymbol{t}$ in the form $\boldsymbol{t}_d(\sigma)$, with $\sigma \in [0,1]$ w.l.o.g., and that $\boldsymbol{t}_d(\sigma)$ is differentiable. For the problem to be well-posed, we assume that:

$$\boldsymbol{t}_d(\sigma) \in \mathcal{T}^*, \quad \forall \sigma \in [0,1],$$

where $\mathcal{T}^* \subset \mathcal{T}$ is the *non-singular task space*, defined as the set of regular and co-regular task space points. The *workspace* $\mathcal{W}$ (a subset of $I\!R^2$ or $I\!R^3$ depending on whether we are considering planar or spatial motions) is populated by obstacles.

In the above hypotheses, the Task-Constrained Motion Planning (TCMP) problem consists in finding a configuration space path $\boldsymbol{q}(s), s \in [s_0, s_1]$, and a surjective, non-decreasing mapping[3] $\sigma(s) : [s_0, s_1] \mapsto [0,1]$ such that:

1) $\boldsymbol{t}(s) = \boldsymbol{f}(\boldsymbol{q}(s)) = \boldsymbol{t}_d(\sigma(s)), \forall s \in [s_0, s_1]$;
2) the robot does not collide with obstacles or with itself.

The planning space for the TCMP problem is

$$\mathcal{C}_{\text{task}} = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{t}_d(\sigma), \text{for some } \sigma \in [0,1]\}.$$

The manifold $\mathcal{C}_{\text{task}}$, that we call *task-constrained configuration space*, has naturally the structure of a foliation (see Fig. 1), whose generic *leaf* is defined as

$$\mathcal{L}(\sigma) = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{t}_d(\sigma)\}.$$

Clearly, the existence of a solution to the TCMP problem is determined by the obstacle placement, and in particular by the connectivity of $\mathcal{C}_{\text{task}} \cap \mathcal{C}_{\text{free}}$, the portion of the free configuration space that is compatible with the task path constraint. Depending on the application, an initial joint configuration $\boldsymbol{q}(0)$ such that $\boldsymbol{t}(0) = \boldsymbol{t}_d(0)$ may or may not be assigned. For example, the first is the case when the task is assigned on the basis of sensory information gathered at the current robot posture. In other situations, the determination of $\boldsymbol{q}(0)$ is left to the planning algorithm. The first version of the problem is clearly more constrained (and thus easier to solve, provided that a solution exists) than the second. In the rest of the paper, we assume that $\boldsymbol{q}(0)$ is not assigned.

See the concluding section for additional comments on the nature of task constraints included in our formulation.

[1] If $s = t$, a trajectory is directly planned rather than a path.

[2] Some elements of $\boldsymbol{J}$, however, are not partial derivatives, due to the embedded nonholonomic constraint.

[3] We use different parametrizations for $\boldsymbol{t}_d$ and $\boldsymbol{q}$ to take advantage of the possibility of performing self-motions or backward motions; see the next section.

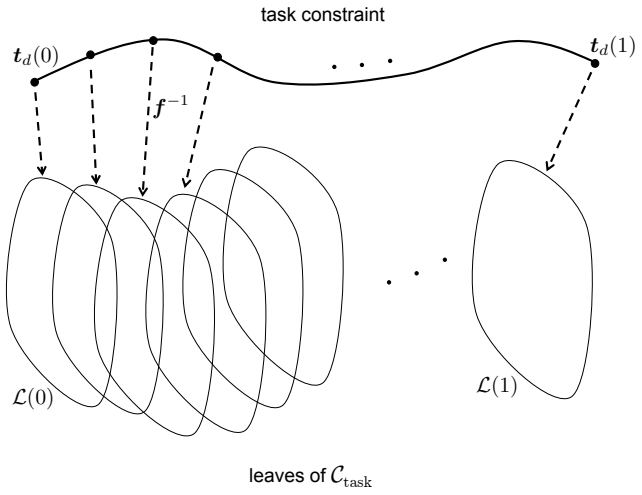


Fig. 1. The task-constrained configuration space $\mathcal{C}_{\rm task}$ is a foliation.

IV. Motion Generation

Our control-based planner relies on the following motion generation scheme:

$$\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q}_a) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \tilde{\boldsymbol{v}} \quad (6)$$

$$\tilde{\boldsymbol{v}} = \boldsymbol{J}^\dagger(\boldsymbol{q})(\boldsymbol{t}_d' + k\,\boldsymbol{e}_t) + (\boldsymbol{I} - \boldsymbol{J}^\dagger(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}))\tilde{\boldsymbol{w}}, \quad (7)$$

where $\boldsymbol{J}^\dagger$ is the pseudoinverse of the task Jacobian $\boldsymbol{J}$, k is a positive gain, $\boldsymbol{e}_t = \boldsymbol{t}_d - \boldsymbol{t}$ is the task error, $\boldsymbol{I} - \boldsymbol{J}^\dagger\boldsymbol{J}$ is the orthogonal projection matrix in the null space of $\boldsymbol{J}$, and $\tilde{\boldsymbol{w}}$ is an arbitrary n-vector that acts as a *residual* input. Note that $\boldsymbol{J}^\dagger = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}$ when $\boldsymbol{J}$ is full row rank; this assumption is always satisfied because our planner discards singular configurations (see the EXTEND function in Sect. V).

Substituting the above expression of $\tilde{\boldsymbol{v}}$ in (5) one obtains $\boldsymbol{e}_t' = -k\,\boldsymbol{e}_t$, i.e., asymptotically stable[4] tracking of the desired task path. In principle, by solving the differential system (6–7) starting from $\mathcal{L}(0)$ with different choices of the residual input $\tilde{\boldsymbol{w}}(s)$, $s \in [s_0, s_1]$, one may generate all configuration paths satisfying the task constraint. A numeric solver can be used to actually perform the integration.

Starting from a generic leaf of $\mathcal{C}_{\rm task}$, forward integration of (6–7) allows to obtain a configuration path that starts from the current leaf and traverses subsequent leaves. However, taking advantage of the different parameterizations of the task space and the configuration space paths, one may also perform a motion on the same leaf (self-motion) or even move backwards from the current leaf towards previous leaves. In formulas:

$$\boldsymbol{t}_d' = \frac{d\boldsymbol{t}_d}{ds} = \frac{d\boldsymbol{t}_d}{d\sigma} \cdot \frac{d\sigma}{ds}$$

where

$$\frac{d\sigma}{ds} = \begin{cases} 1 & \text{forward motion} \\ 0 & \text{self-motion} \\ -1 & \text{backward motion} \end{cases}$$

[4]Asymptotic stability is essential in reducing the drift that is invariably associated to a numerical integration of (6–7).

Since system (6–7) is symmetric, any backward motion can be reversed to enforce the condition that the task path parameter σ does not decrease as the configuration path parameter s increases. In practice, this is simply realized by redefining s along the subpath so as to increase from the previous leaf to the current leaf.

V. Control-Based TCMP

Our control-based randomized planner builds a Rapidly-exploring Random Tree (RRT, see [12], [18]) in the task-constrained configuration space $\mathcal{C}_{\rm task}$ to search for a collision-free path. For the construction of the tree, we make use of samples of the desired task path $\boldsymbol{t}_d(\sigma)$; in particular, denoting by $\{\sigma_1 = 0, \sigma_2, \ldots, \sigma_{N-1}, \sigma_N = 1\}$ an equispaced sequence of N path parameter values, let $\boldsymbol{t}_k = \boldsymbol{t}_d(\sigma_k)$ (we drop the d subscript for compactness). The tree edges are collision-free subpaths obtained by applying the motion generation scheme (6–7) starting from the tree nodes, that are configurations in $\mathcal{C}_{\rm task}$; in particular, each node has an associated leaf index k ($k = 1, \ldots, N$) that identifies the particular leaf $\mathcal{L}_k = \mathcal{L}(\sigma_k)$ on which it lies.

The core of the planner is the EXTEND procedure described in pseudocode in Fig. 2. Given a tree T and a configuration $\boldsymbol{q}_{\rm rand}$, EXTEND first finds the node $\boldsymbol{q}_{\rm near}$ in T that is closest to $\boldsymbol{q}_{\rm rand}$ (line 1); call k the associated leaf index. T is then extended from $\boldsymbol{q}_{\rm near}$ using the motion generation scheme (6–7) to perform (respectively lines 2, 6, 10):

- a *forward motion*, that produces a subpath leading to a configuration $\boldsymbol{q}_{\rm fw}$ on $\mathcal{L}_{k+1}$;
- a *self-motion*, that produces a subpath leading to a configuration $\boldsymbol{q}_{\rm self}$ on $\mathcal{L}_k$;
- a *backward motion*, that produces a subpath leading to a configuration $\boldsymbol{q}_{\rm bw}$ on $\mathcal{L}_{k-1}$.

The task constraint is exactly satisfied along these subpaths in view of the use of (6–7). Whenever a singular configuration is generated during FM, BM or SM, the integration stops and $\boldsymbol{q}_{\rm fw}$, $\boldsymbol{q}_{\rm self}$ or $\boldsymbol{q}_{\rm bw}$ are set to a predefined singular value.

Different choices of the residual input $\tilde{\boldsymbol{w}}$ are possible for generating motions with (6–7). Among the various possibilities we mention: (1) a random choice with a bounded or constant norm; (2) a random choice within a finite set of motion primitives; (3) an optimal choice within a finite set of motion primitives with respect to a predefined criterion, such as the distance to $\boldsymbol{q}_{\rm rand}$ (other options are the manipulability index [19] and the task compatibility index [20]).

For each generated subpath, the VALID procedure is called to verify that the terminal configuration ($\boldsymbol{q}_{\rm fw}$, $\boldsymbol{q}_{\rm self}$ or $\boldsymbol{q}_{\rm bw}$) is neither singular[5] nor in collision with the obstacles (lines 3, 7 and 11). If this test is passed, the subpath itself is checked for collision. In the negative case, the terminal configuration and the subpath are inserted in T as a node and an edge, respectively. In particular, a backward motion

[5]Discarding singular configurations does not hinder the possibility of finding solutions. In fact, in the hypotheses of the TCMP problem, whenever a singular configuration is generated there always exist (arbitrarily close) regular configurations realizing the same task value, that the planner can find and use to build a solution.

```
EXTEND(T, q_rand)
1   q_near ← NEAREST_NODE(T, q_rand);
2   q_fw ← FM(q_near);
3   if VALID(q_fw) and FREE_PATH(q_near,q_fw)
4     T.add_node(q_fw);
5     T.add_edge(q_near, q_fw);
6   q_self ← SM(q_near);
7   if VALID(q_self) and FREE_PATH(q_near,q_self)
8     T.add_node(q_self);
9     T.add_edge(q_near, q_self);
10  q_bw ← BM(q_near);
11  if VALID(q_bw) and FREE_PATH(q_near,q_bw)
12    T.add_node(q_bw);
13    T.add_edge(q_bw, q_near);
14  return (q_fw, q_self, q_bw);
```

Fig. 2. The EXTEND procedure for growing the exploration tree.

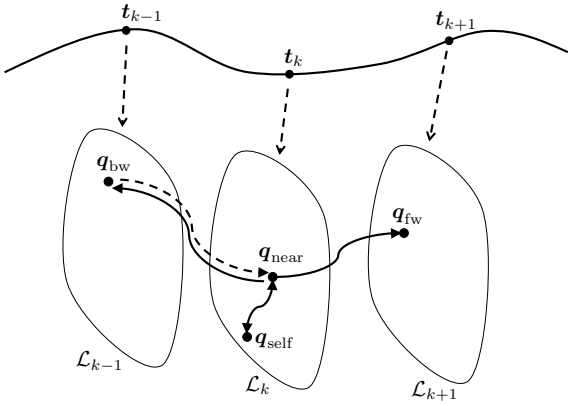


Fig. 3. An extension step from $\boldsymbol{q}_{\rm near}$. Note that the direction of the subpath from $\boldsymbol{q}_{\rm near}$ to $\boldsymbol{q}_{\rm bw}$ is actually reversed before storing it as an edge in T.

is reversed and stored as an edge directed from $\boldsymbol{q}_{\rm bw}$ to $\boldsymbol{q}_{\rm near}$ (see the previous section), while a self-motion generates a bidirectional edge between $\boldsymbol{q}_{\rm near}$ and $\boldsymbol{q}_{\rm self}$ (i.e., an edge which can be traversed in both directions).

The CONTROL_BASED planner that uses the EXTEND procedure is described in Fig. 4. A random collision-free configuration $\boldsymbol{q}_{\rm init}$ lying on $\mathcal{L}_1$ is first generated (line 2) by calling RAND_CONF with argument $\boldsymbol{t}_1$; essentially, this function selects $n_q - n_t$ configuration coordinates at random and then solves for the remaining n_t coordinates so as to realize the assigned task value. The configuration $\boldsymbol{q}_{\rm init}$ is used to initialize the tree T (line 3). The algorithm then enters the main cycle (lines 4–11) in which it tries to extend T in order to connect $\mathcal{L}_1$ to $\mathcal{L}_N$ with a collision-free path in $\mathcal{C}_{\rm task}$. Each iteration proceeds by calling first RAND_CONF to generate a random configuration $\boldsymbol{q}_{\rm rand}$ in $\mathcal{C}$ (line 6), and then EXTEND (line 7) to grow the tree towards $\boldsymbol{q}_{\rm rand}$.

If EXTEND places a node on either $\mathcal{L}_1$ or $\mathcal{L}_N$, T is searched for an optimal path[6] from $\mathcal{L}_1$ to $\mathcal{L}_N$ (line 9). In view of the structure of the algorithm, it is easy to show

[6]Particularly in the early stages of planning, EXTEND may place a node on $\mathcal{L}_1$ before a connection to $\mathcal{L}_N$ has been created; in this case, a path from $\mathcal{L}_1$ to $\mathcal{L}_N$ does not exist yet. On the other hand, when EXTEND places a node on $\mathcal{L}_N$, a connection exists between $\mathcal{L}_1$ and $\mathcal{L}_N$, but it is not guaranteed that the path parameter s is monotonically increasing along this path. In both these cases, the path search will simply fail and the planning phase will resume.

```
CONTROL_BASED
1   i ← 0;
2   q_init ← RAND_CONF(t_1);
3   T.init(q_init);
4   repeat
5     i ← i + 1;
6     q_rand ← RAND_CONF;
7     (q_fw, q_self, q_bw) ← EXTEND(T, q_rand);
8     if q_fw ∈ L_N OR if q_bw ∈ L_1
9       P=SHORTEST_PATH(L_1,L_N, T);
10    else P=NULL;
11  until P!=NULL or i = MAX_IT
12  return P;
```

Fig. 4. The CONTROL_BASED algorithm for solving the TCMP problem.

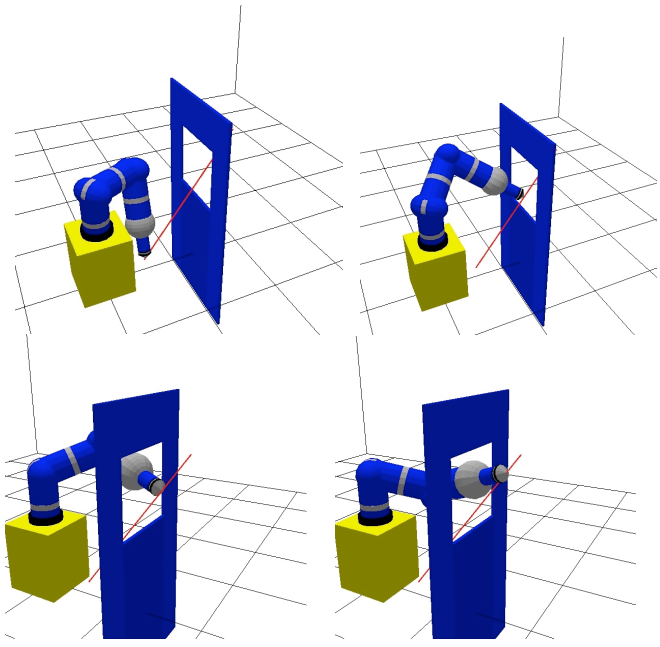
that path search can be performed by considering a single start (the last generated node, lying on either $\mathcal{L}_N$ or $\mathcal{L}_1$) and multiple goals (all the nodes on the other extremal leaf $\mathcal{L}_1$ or $\mathcal{L}_N$). The loop is repeated until a solution path can be extracted from T or the maximum number of attempts MAX_IT has been reached.

VI. Results

The CONTROL_BASED algorithm for solving the TCMP problem has been implemented in Move3D [21], a motion planning software platform developed at LAAS-CNRS. In this section, we present some preliminary results obtained by applying the proposed planning method to (1) a fixed-base manipulator and (2) a robot with a free-flying base. Since these systems are not subject to nonholonomic constraints, the motion generation scheme actually reduces in both cases to (7), with $\tilde{\boldsymbol{v}} = \boldsymbol{q}'$. All the reported experiments have been performed on a Dual Core Pentium 4 running at 3GHz under Linux Red Hat 8. Videos clips are available at `http://www.dis.uniroma1.it/labrob/research/TCMP.html`.

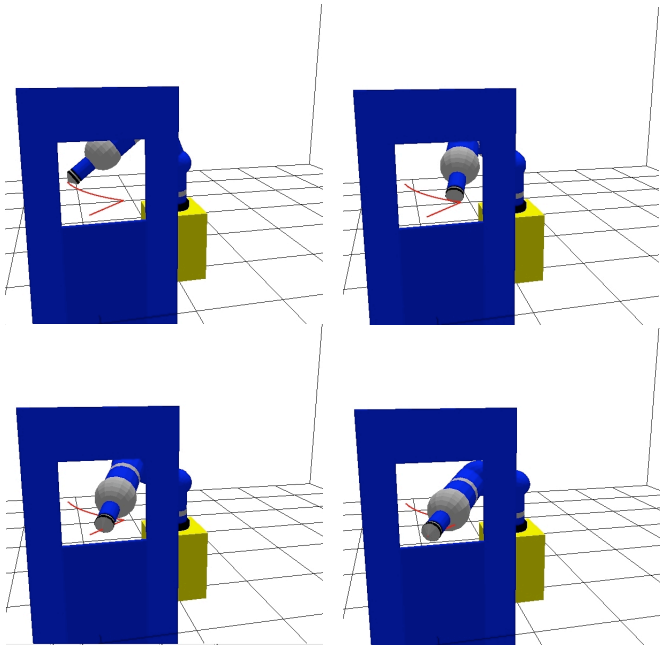
The first two experiments have been performed on the DLR Light Weight Robot, a dextrous manipulator with 7 revolute joints. The task is to follow an assigned path for the position of the end-effector. In both cases, the last three degrees of freedom (the wrist) are blocked; the degree of redundancy is therefore equal to 1. To better evaluate the practical improvement introduced by the proposed strategy, we have solved two planning problems of increasing difficulty and compared the results obtained by the CONTROL_BASED planner and the RRT_LIKE planner of [4], an RRT-based strategy that uses a linear local planner. The reported results (averaged on ten realizations of the planning process) were produced with a random generation of $\tilde{\boldsymbol{w}}$ and by using a simple Eulero algorithm, with integration step $\Delta s = 0.0025$, to generate motion with (7). Euclidean distance in $\mathcal{C}$ is used to identify the configuration $\boldsymbol{q}_{\rm near}$ in T closest to $\boldsymbol{q}_{\rm rand}$.

The first planning experiment is illustrated in Fig. 5. The assigned end-effector path is a line segment, of length equal to 120 cm, passing through the window placed in front of the manipulator. The first two rows of the table in Fig. 5 show the results obtained with CONTROL_BASED and RRT_LIKE with $N = 10$ task path samples (with this choice, the distance between the samples along the path is about 13 cm). The values of both the mean and the maximum tracking error



Planner	Mean Error	Max Error	# nodes
CONTROL_BASED	0.0168 cm	0.0754 cm	91.7
RRT_LIKE	0.6649 cm	2.9790 cm	68.2
RRT_LIKE ($N = 100$)	0.1149 cm	1.5371 cm	1179.2

Fig. 5. First planning experiment on the DLR Light Weight Robot: comparison between CONTROL_BASED with $N = 10$ path samples and RRT_LIKE [4] with $N = 10$ and $N = 100$.



Planner	Mean Error	Max Error	# nodes
CONTROL_BASED	0.0098 cm	0.0436 cm	834.6
RRT_LIKE ($N = 100$)	0.1479 cm	0.8608 cm	1058.9

Fig. 6. Second planning experiment on the DLR Light Weight Robot: comparison between CONTROL_BASED with $N = 10$ path samples and RRT_LIKE [4] with $N = 100$.

obtained with CONTROL_BASED are smaller by almost two orders of magnitude than those with RRT_LIKE. This higher accuracy is achieved without a significant increase in the size (number of nodes) of the tree. In fact, the average running time was less than 1 s for both planners.

In principle, the tracking performance of the RRT_LIKE planner can be improved by increasing the number of path samples. The third row of the table in Fig. 5 reports the results obtained with $N = 100$. These data show a limited improvement in the tracking error (still larger than the error achieved by CONTROL_BASED with $N = 10$) at the price of a severe performance degradation in terms of tree size and running time (now about 17 s).

In the second planning problem (see Fig. 6) the path and the window were moved further right w.r.t. the base of the manipulator. This reduces the size of $\mathcal{C}_{\text{task}}$, generating a narrow passage in $\mathcal{C}_{\text{task}} \cap \mathcal{C}_{\text{free}}$ that complicates the search of a collision-free path satisfying the task constraint. The reported results have been obtained using $N = 10$ path samples for CONTROL_BASED and $N = 100$ for RRT_LIKE. The path is about 85 cm long, so that the distance between the path samples is about 9 cm for CONTROL_BASED and 0.9 cm for RRT_LIKE. As in the first experiment, the control-based strategy gives better results in terms of tracking accuracy, which is comparable to that of the first experiment. The final size of the trees is similar, and in fact the running time was about 15 s for both planners.

The last planning scene is shown in Fig. 7. A 7-dof robot consisting of a 4-dof linkage mounted on a 3-dof free-flying base has to move its end-point along a 33 m path inside a pipe of variable radius. The averaged results have been obtained with a random choice of the vector $\tilde{\boldsymbol{w}}$ and $N = 10$ path samples. The Eulero integration step used to generate motion with (7) has been set to $\Delta s = 0.01$. The mean and maximum errors were respectively 0.0386 cm and 0.3223 cm, while the number of nodes in the tree was 99.6. The average running time was about 2 s.

Overall, our numerical results indicate that the CONTROL_BASED planner outperforms the RRT_LIKE planner of [4] both in terms of tracking error and complexity. Even better (in principle, arbitrary) accuracy can be obtained by using a higher-order integration algorithm, without increasing the size of the tree. Therefore, the running time of CONTROL_BASED nicely scales with the complexity of the planning problem, independently of the accuracy required by the task tracking. Moreover, it may be expected that appropriate heuristics for choosing $\tilde{\boldsymbol{w}}$ will further improve the efficiency of CONTROL_BASED.

VII. Conclusion

We have presented a control-based randomized technique for solving the Task-Constrained Motion Planning problem in redundant robotic systems, possibly subject to nonholonomic

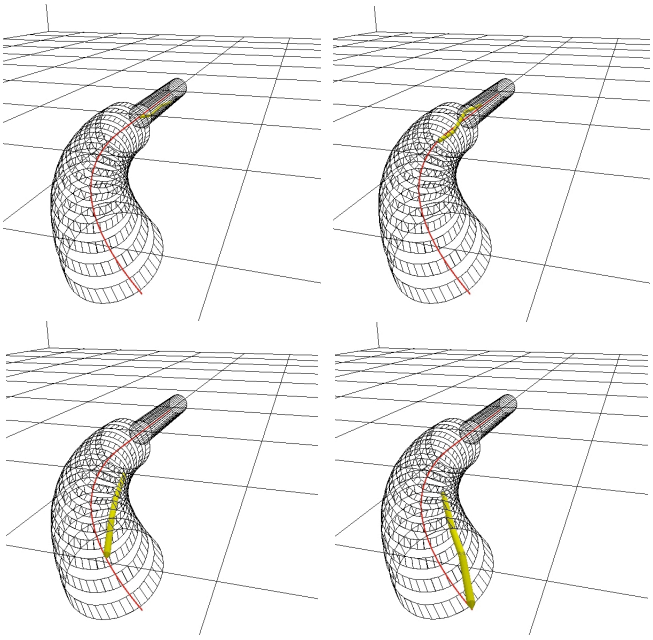


Fig. 7. Third planning experiment: a 7-dof robot with free-flying base moving its end-point along an inspection path inside a variable radius pipe.

constraints. At the core of our method is a motion generation scheme that guarantees continued satisfaction of the task constraint, allowing to achieve accurate execution of the desired task without increasing the size of the roadmap. Preliminary results are very encouraging and confirm the performance improvement obtained with the proposed technique.

Note that we have only considered the case of a *parameterized* (e.g., timed) task constraint $\boldsymbol{t}_d(s)$, and in fact the presented planner takes advantage of the foliation induced by the parameterization. Non-parameterized constraints, such as keeping a grasped object at a fixed orientation or maintaining loop closure in a closed-chain manipulator, are not considered in this paper because they encode restrictions rather than tasks: to create an actual motion requirement, a start and a goal configuration must be assigned separately. Therefore, planning under non-parameterized constraints should be simply indicated as Constrained Motion Planning (CMP). However, the motion generation scheme (6–7) is also beneficial in CMP problems, because it can be effectively used in conjunction with any control-based randomized planner.

Future work will address several points, among which:

- the choice of metrics for $\mathcal{C}_{\mathrm{task}}$ and its impact on performance;
- the design of effective heuristics for choosing $\tilde{\boldsymbol{w}}$ in the motion generation scheme;
- the application of the planner to more complex robotic structures, such as humanoids;
- the extension of the proposed method to the case of inequality task constraints.

Finally, on the basis of some preliminary analysis as well as of the obtained results, we conjecture that the probabilistic completeness of the RRT algorithm is preserved with our planning scheme, at least for the case of random choice of the residual inputs $\tilde{\boldsymbol{w}}$, despite the fact that the construction of the tree takes place in the task constrained space $\mathcal{C}_{\mathrm{task}}$. Future work will also be aimed at devising a formal proof of this conjecture.

REFERENCES

[1] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial", *J. of Intelligent and Robotic Systems*, vol. 3, pp. 201–212, 1990.

[2] D. P. Martin, J. Baillieul, and J.M. Hollerbach (1989), "Resolution of kinematic redundancy using optimization techniques," *IEEE Trans. on Robotics and Automation*, vol. 5, pp. 529–533, 1989.

[3] S. Chiaverini, G. Oriolo and I. Walker, "Chapter 11: Kinematically redundant manipulators," in *Handbook of Robotics*, O. Khatib and B. Siciliano (Eds), Springer, 2009.

[4] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," *2002 IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1657–1662, 2002.

[5] G. Oriolo, C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," *2005 IEEE Int. Conf. on Robotics and Automation*, pp. 2166–2172, 2005.

[6] M. Stilman, "Task constrained motion planning in robot joint space," *2007 IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 3074–3081, 2007.

[7] L. Han and N. Amato, "A kinematic-based probabilistic roadmap method for closed chain systems," *4th Int. Work. on Algorithmic Foundations of Robotics*, pp. 233–246, 2000.

[8] J. Yakey, S. M. LaValle and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, pp. 951–958, 2001.

[9] J. Cortes, T. Simèon, and J.P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using PRM methods," *2002 IEEE Int. Conf. on Robotics and Automation*, pp. 2141–2146, 2002.

[10] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms and Implementations*, MIT Press, Cambridge, MA, 2005.

[11] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[12] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[13] J. Barraquand, J.C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, pp. 121–155, 1993.

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

[15] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modeling, Planning and Control*, Springer, London, 2009.

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

[17] J. Burdick, "On the inverse kinematics of redundant manipulators: Characterization of the self motion manifolds," *1989 IEEE Int. Conf. on Robotics and Automation*, pp. 264–270, 1989.

[18] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., Computer Science Dept., Iowa State University, 1998.

[19] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley, 1991.

[20] S. Chiu, "Task compatibility of manipulator postures," *The International Journal of Robotics Research*, vol. 7, no. 5, pp. 13-21, 1988.

[21] T. Simeon, J.-P. Laumond, and F. Lamiraux, "Move3d: A generic platform for path planning," *4th Int. Symp. on Assembly and Task Planning*, pp. 25–30, 2001.