DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

A kinodynamic approach to task-constrained motion planning

Giuseppe Oriolo
Pietro Peliti
Marilena Vendittelli

Technical Report n. 4, 2009

A kinodynamic approach to task-constrained motion planning

Giuseppe Oriolo Pietro Peliti Marilena Vendittelli
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Ariosto 25, 00185 Roma, Italy
{oriolo,vendittelli}@dis.uniroma1.it, pepope@hotmail.it

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 kinodynamic approach based on a motion generation scheme that guarantees continued satisfaction of such constraint. The resulting randomized planner allows to achieve accurate execution of the desired task without increasing the complexity 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.

Keywords: redundant robots, kinodynamic planning, task space constraints.

1 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. Redundant robotic systems, such as humanoids and mobile manipulators, posses 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 workspace obstacles or with parts of itself (self-collision). In the following, this problem is referred to as Task-Constrained Motion Planning (TCMP).

Earlier researchers 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 kinematically redundant fixed-base manipulators with 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 task space constraints is also proposed, and in [7]. A related problem is motion planning for closed kinematics chains, in which the closure condition may be considered as a task constraint; randomized techniques for this problem have been presented in [8, 9, 10].

All the above randomized planning techniques are based on the idea of sampling and inverting the task constraint to build a roadmap consisting of task-constrained 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 complexity of the roadmap and, as a consequence, of the time needed to solve the planning problem.

In this paper, a solution is proposed to the TCMP problem based on the principle of kinodynamic planning [11, 12]. While this technique has been typically used to comply with nonholonomic constraints, we apply it to comply with a parametric holonomic constraint that represents the desired task constraint. Using a motion generation scheme that guarantees continued

satisfaction of such constraint, we develop an RRT-based planner that allows to achieve accurate execution of the desired task without increasing the complexity of the roadmap. The proposed technique applies to general robotic systems, that may also be subject to nonholonomic constraints.

The paper is organized as follows. Sect. 2 presents some background material on task kinematics and redundancy concepts. The TCMP problem is formulated in Sect. 3, while our motion generation scheme is introduced in Sect. 4. The kinodynamic planner for solving the TCMP problem is presented in Sect. 5. Numerical results are given in Sect. 6.

2 Task Kinematics and Redundancy

Consider a robotic system whose configuration $\boldsymbol{q}$ takes value in an n_q-dimensional configuration space $\mathcal{Q}$. For convenience, $\boldsymbol{q}$ is partitioned 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; accordingly, it is $\mathcal{Q} = \mathcal{Q}_a \times \mathcal{Q}_b$. For the sake of generality, we allow that $\boldsymbol{q}_a$ is subject to n_c nonholonomic constraints of the Pfaffian form

$$\boldsymbol{A}(\boldsymbol{q}_a)\dot{\boldsymbol{q}}_a = 0,$$

which may be rewritten in a geometric form [13] as

$$\boldsymbol{A}(\boldsymbol{q}_a)\boldsymbol{q}_a' = 0, \tag{1}$$

where $\boldsymbol{q}_a' = d\boldsymbol{q}_a/ds$ and s is a path parameter (if $s = t$, a trajectory is planned rather than a path). This description fits most robotic systems, including fixed-base manipulators, wheeled or legged mobile robots, and mobile manipulators.

The kinematic model of the system is expressed in geometric form as

$$\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{q}_a' \\ \boldsymbol{q}_b' \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q}_a)\tilde{\boldsymbol{v}}_a \\ \tilde{\boldsymbol{v}}_b \end{pmatrix} \tag{2}$$

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 motion of the $\boldsymbol{q}_a$ coordinates in $\mathcal{Q}_a$ is locally constrained, 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 inputs are *geometric* velocities, i.e., tangent vectors. The driftless system (2) is controllable in view of the nonholonomy of the constraints (1).

Consider now a set of task coordinates $\boldsymbol{t}$ taking values in an n_t-dimensional space $\mathcal{T}$. The task coordinates are related to the configuration coordinates by the kinematic map

$$\boldsymbol{t} = \boldsymbol{f}(\boldsymbol{q}). \tag{3}$$

At the differential level, we have

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

where $\boldsymbol{J_f}(\boldsymbol{q}) = d\boldsymbol{f}/d\boldsymbol{q}$, and using (2)

$$\boldsymbol{t}' = \begin{pmatrix} \boldsymbol{J_{f,a}}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q}_a) & \boldsymbol{J_{f,b}}(\boldsymbol{q}) \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{v}}_a \\ \tilde{\boldsymbol{v}}_b \end{pmatrix} = \boldsymbol{J}(\boldsymbol{q})\tilde{\boldsymbol{v}}. \tag{4}$$

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*[1] in the following, maps the admissible instantaneous motions of the robot to the geometric velocities of the task coordinates [14]. Typical tasks concern manipulation (end-effector position and/or orientation) or perception (sensor pose or even features in the sensing space, as in visual servoing).

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

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 $\bar{\boldsymbol{q}} = f^{-1}(\bar{\boldsymbol{t}})$ of a certain 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 [15]. In particular, task points of the first group include *regular* points and *coregular* points (avoidable singularities), whereas the second group consists of *singular* points (unavoidable singularities).

3 The TCMP Problem

Consider a robotic system in the form (2) that is kinematically redundant for the task of interest, related to the configuration variables by eq. (3).

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

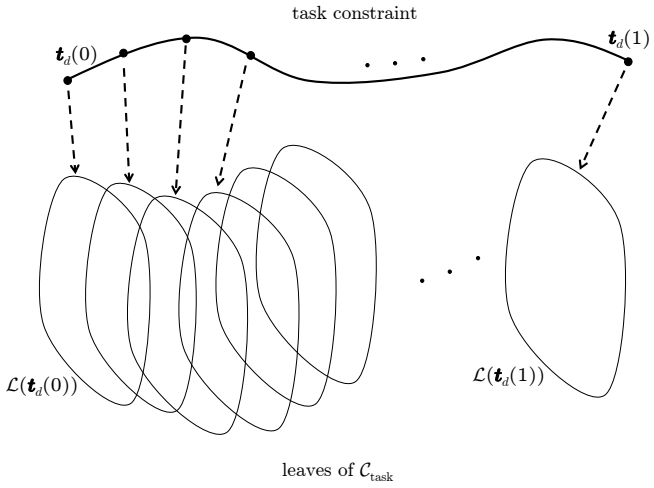


Figure 1: The task-constrained configuration space $\mathcal{C}_{\text{task}}$ is a foliation

Assume that a desired path is assigned for the task variables $\boldsymbol{t}$ in the form $\boldsymbol{t}_d(s)$, with $s \in [0,1]$ the path parameter, and that $\boldsymbol{t}_d(s)$ is differentiable. For the problem to be well-posed, we assume that:

$$\boldsymbol{t}_d(s) \in \mathcal{T}_{\text{reg}}, \quad \forall s \in [0,1],$$

where $\mathcal{T}_{\text{reg}} \subset \mathcal{T}$ is the *regular task space*, defined as the set of regular task space points (whose inverse image does not contain singular 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.

The Task-Constrained Motion Planning (TCMP) problem consists in finding a path $\boldsymbol{q}(s)$ in the configuration space such that:

1. $\boldsymbol{t}(s) = \boldsymbol{t}_d(s)$, $\forall s \in [0,1]$;
2. the robot does not collide with obstacles or with itself.

Note the following points:

- The planning space for the TCMP problem is

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

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

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

- A solution to the TCMP problem may exist or not depending on the obstacle placement, and in particular on 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 trajectory is planned on the basis of sensory information gathered at the current robot posture. On the other hand, the determination of $\boldsymbol{q}(0)$ will be typically left to the planning algorithm when the task is assigned off-line. 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.

4 Motion Generation

The kinodynamic planner that is presented in this paper relies on the following motion generation scheme:

$$\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{q}_a) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \tilde{\boldsymbol{v}} \tag{5}$$

$$\tilde{\boldsymbol{v}} = \boldsymbol{J}^{\dagger}(\boldsymbol{q})(\boldsymbol{t}_d' + k\,\boldsymbol{e_t}) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}))\tilde{\boldsymbol{w}}, \tag{6}$$

where $\boldsymbol{J}^{\dagger}$ is the pseudoinverse of $\boldsymbol{J}$, 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. Note that $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}$, because $\boldsymbol{J}$ is always full row rank in the hypotheses of the TCMP problem.

Substituting the above expression of $\tilde{\boldsymbol{v}}$ in (4) one obtains $\boldsymbol{e}_t' = -k\,\boldsymbol{e}_t$, i.e., asymptotically stable[2] tracking of the desired task path. Therefore, as $\tilde{\boldsymbol{w}}$ varies, the solution of the differential equation (5–6) initialized on $\mathcal{C}_{\text{task}}$ provides all configuration paths satisfying the task constraint.

Starting from a generic leaf of $\mathcal{C}_{\text{task}}$, integration of (5–6) allows to obtain a configuration path lying on the subsequent leaves. One may also move backwards in s (i.e., on the previous leaves), by using $-\boldsymbol{t}_d'$ in place of $\boldsymbol{t}_d'$, or move on the same leaf, setting $\boldsymbol{t}_d' = 0$. A numeric solver can be used to actually perform the integration.

5 TCMP via Kinodynamic Planning

This section describes the strategy used to search the task-constrained configuration space $\mathcal{C}_{\text{task}}$ of the redundant system for a collision-free path. The

[2] Asymptotic stability is essential in reducing the drift that is invariably associated to a numerical integration of (5–6).

```
TCMP_KINO
1    i ← 0;
2    q_init ← INV_KIN(t_1);
3    T.init(q_init);
4    repeat
5      i ← i + 1;
6      q_rand ← RAND_CONF;
7      (t_new, q_new) ← EXTEND(T, q_rand);
8      if t_new!=NULL
9        if t_new = t_N
10         L_goal = L_1;
11       else
12         L_goal = L_N;
13       P=SHORTEST_PATH(q_new, L_goal, T);
14     else P=NULL;
15   until P!=NULL or i = MAX_IT
16   if P!=NULL return P;
17   else return FAILURE;
```

Figure 2: The TCMP_KINO algorithm.

main ingredient is a randomized kinodynamic planner based on the construction of a Rapidly-exploring Random Tree (RRT) evolving in $\mathcal{C}_{\text{task}}$ [12, 16, 17]. For the construction of the tree, we make use of samples of the desired task path $\boldsymbol{t}_d(s)$; in particular, denoting by $\{s_1 = 0, s_2, \ldots, s_N = 1\}$ an equispaced sequence of path parameter values, let $\boldsymbol{t}_k = \boldsymbol{t}_d(s_k)$ (note that we drop the d subscript for compactness). The tree edges are collision-free paths obtained by applying the motion generation scheme (5–6) starting from the tree nodes, that lie on the leaves $\mathcal{L}_k = \mathcal{L}(\boldsymbol{t}_k)$, for $k = 1, \ldots, N$.

A pseudocode description of the algorithm is given in Fig. 2. A collision-free configuration $\boldsymbol{q}_{\text{init}}$ lying on $\mathcal{L}_1$ is first generated by calling the function INV_KIN with argument $\boldsymbol{t}_1$ (line 2). This configuration is then used to initialize the tree T (line 3). On line 4 the algorithm enters the main cycle (lines 4–15) in which it tries to extend T so as to connect $\mathcal{L}_1$ to $\mathcal{L}_N$ with a collision-free path in $\mathcal{C}_{\text{task}}$. Each iteration proceeds by first generating a random configuration $\boldsymbol{q}_{\text{rand}}$ (line 6) and then calling the EXTEND procedure (line 7) to extend the tree towards $\boldsymbol{q}_{\text{rand}}$. If $\mathcal{L}_1$ or $\mathcal{L}_N$ are reached by EXTEND, the procedure returns the corresponding configuration. In this case, T is searched for an optimal path from $\mathcal{L}_1$ to $\mathcal{L}_N$. The loop is repeated until either a path is found from $\mathcal{L}_1$ to $\mathcal{L}_N$ or the maximum number of attempts MAX_IT has been reached.

The core of the algorithm is the EXTEND procedure shown in Fig. 3. Given the current T and the configuration $\boldsymbol{q}_{\text{rand}}$, the procedure determines the nearest node $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{rand}}$ in T and the associated leaf index k (line 1). The tree T is then extended using the motion generation scheme (5–6) from $\boldsymbol{q}_{\text{near}}$, that lies on $\mathcal{L}_k$, to perform a *forward motion* towards $\mathcal{L}_{k+1}$, a *backward*

```
EXTEND(T, q_rand)
1    (q_near, k) ← 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      if t_{k+1} = t_N return t_{k+1}, q_fw;
7    q_bw ← BM(q_near);
8    if VALID(q_bw) and FREE_PATH(q_near,q_bw)
9      T.add_node(q_bw);
10     T.add_edge(q_bw, q_near);
11     if t_{k-1} = t_1 return t_{k-1}, q_bw;
12   q_self ← SM(q_near);
13   if VALID(q_self) and FREE_PATH(q_near,q_self)
14     T.add_node(q_self);
15     T.add_edge(q_near, q_self);
16   return NULL;
```

Figure 3: The EXTEND procedure.

motion towards $\mathcal{L}_{k-1}$ and a *self-motion* on $\mathcal{L}_k$ (respectively lines 2, 7, 12). In particular, the first is obtained by forward integration of (5–6) on the interval $[s_k, s_{k+1}]$ and produces an edge leading to a configuration $\boldsymbol{q}_{\rm fw}$ on $\mathcal{L}_{k+1}$; the second is obtained by backward integration on the interval $[s_k, s_{k-1}]$ and produces an an edge leading to a configuration $\boldsymbol{q}_{\rm bw}$ on $\mathcal{L}_{k-1}$; and the third is obtained by integrating on $[s_k, s_{k+1}]$ with $\boldsymbol{t}' = 0$ and produces an edge leading to a configuration $\boldsymbol{q}_{\rm self}$ on $\mathcal{L}_k$.

Fig. 4 shows the three subpaths generated by the extension of $\boldsymbol{q}_{\rm near}$. Note that the task constraint is satisfied along these paths in view of the use of the motion generation scheme (5–6). The paths are added as edges to T if they are not in collision with the obstacles. The dashed arrow from $\boldsymbol{q}_{\rm bw}$ to $\boldsymbol{q}_{\rm near}$ indicates that the edge is reversed before storing it in T; in fact, the path parameter s should monotonically increase along a valid solution path[3].

Different choices of the vector $\tilde{\boldsymbol{w}}$ are possible in generating forward, backward and self-motions with (5–6). Among the various possibilities we mention: (1) a random choice with an upper bound on the vector 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 (for example, the distance to $\boldsymbol{q}_{\rm rand}$, the manipulability index [18] or the task compatibility index [19]).

The EXTEND procedure terminates if either $\mathcal{L}_N$ or $\mathcal{L}_1$ has been reached (respectively, lines 6 and 11). If this is not the case, the procedure will be called again until the maximum numbers of attempts has been reached.

[3]Here, we are exploiting the fact that system (5–6) is symmetric.

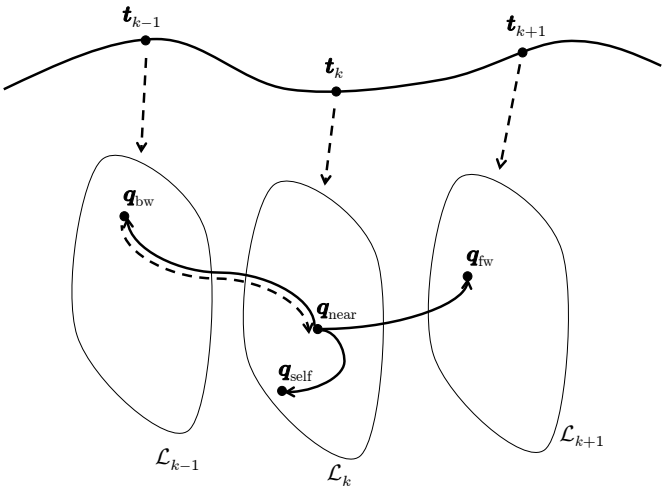


Figure 4: An extension step.

Note that, particularly in the early stages of planning, $\mathcal{L}_1$ can be reached before a connection to $\mathcal{L}_N$ has been created. This event causes the EXTEND procedure to terminate even if a path from $\mathcal{L}_N$ to $\mathcal{L}_1$ does not exist. On the other hand, when a node belonging to $\mathcal{L}_N$ is reached, 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 these cases, the path search will simply fail and the planning phase will resume. It is easy to show that, in view of the structure of the algorithm, path search can be performed by considering a single start (on either $\mathcal{L}_N$ or $\mathcal{L}_1$) and multiple goals (all the nodes on either $\mathcal{L}_1$ or $\mathcal{L}_N$).

6 Results

The algorithm TCMP_KINO has been implemented in Move3D [20], a software platform developed at LAAS-CNRS and dedicated to motion planning[4]. In this section we present preliminary results obtained by applying the proposed planning method to a fixed-base manipulator and to a robot with a free-flying base. All the reported experiments have been performed on a Dual Core Pentium 4 3GHz with 1GB RAM under the Linux Red Hat 8 OS.

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 the experiments, the last three degrees of freedom (the wrist) are blocked; the degree

[4]Move3D is at the origin of the product KineoWorks currently marketed by the company Kineo CAM (www.kineocam.com).

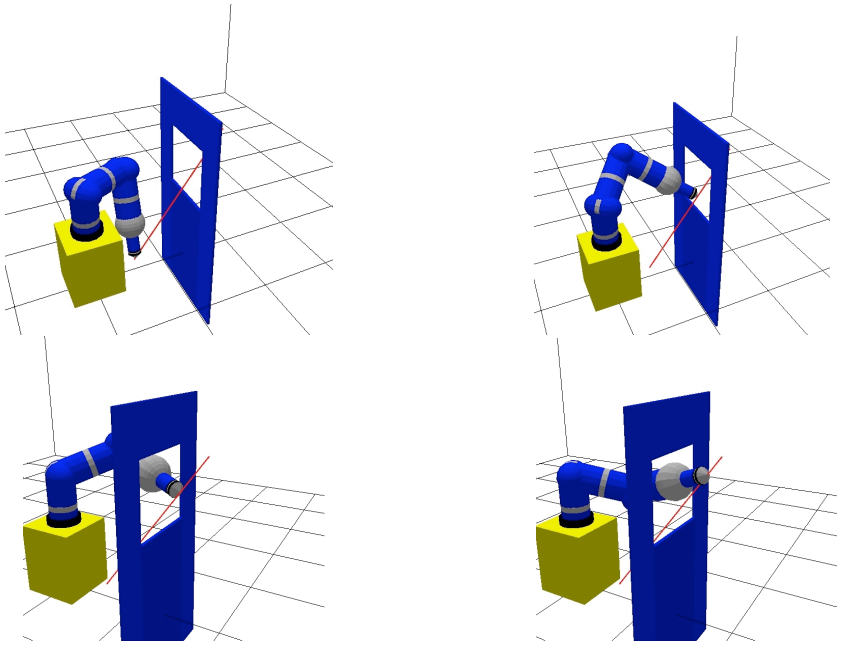
of redundancy is therefore equal to 1. To better evaluate the practical improvement introduced by the proposed strategy, we have solved two planning problems of increasing difficulty and compared the results obtained by using the TCMP_KINO planner and the RRT_LIKE planner developed in [4], an RRT-based strategy that uses a linear local planner. The reported results (averaged on ten realizations of the planning process) have been obtained 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 (5–6).

The first planning experiment is illustrated in Fig. 5. The given end-effector path is a straight segment, of length equal to 120 cm, passing through the window placed in front of the manipulator. The first two lines of the table in Fig. 5 report the data obtained by using $N = 10$ task path samples for both TCMP_KINO and RRT_LIKE. With this choice, the distance between the samples along the path is about 13 cm. Average running time was less than 1 second for both TCMP_KINO and RRT_LIKE. The values of both the mean and the maximum tracking error obtained with the kinodynamic approach are smaller by almost two orders of magnitude than those obtained by using an RRT planning strategy with a linear local planner. This improvement in accuracy is obtained without a significant increase in the complexity (number of nodes) of the tree.

The performance of the RRT_LIKE planner in terms of the tracking error can be improved by increasing the number of path samples. The third line of the table in Fig. 5 reports the results obtained with $N = 100$. These data show that the limited improvement in the tracking error (that is however larger than the error obtained by using TCMP_KINO with $N = 10$) is balanced by a severe degradation of the planner performance in terms of tree complexity and running time (about 17 sec).

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 measure 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 by using $N = 10$ task path samples for TCMP_KINO 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 TCMP_KINO and 0.9 cm for RRT_LIKE. As in the first experiment, the kinodynamic strategy gives better results in terms of tracking accuracy, which is comparable to that of the first experiment. The similar complexity of the final trees (strictly related to the running time, which was about 15 sec for both planners) is due to the intrinsic difficulty of the problem. Comparison with the first experiment shows that the complexity of the tree built by TCMP_KINO depends only on the difficulty of the planning problem. It is expected, however, that the use of heuristics for choosing $\tilde{w}$ will further improve the performance of TCMP_KINO in terms of tree complexity.

The last experimental scene is reported in Fig. 7. A 4-dof robot on a

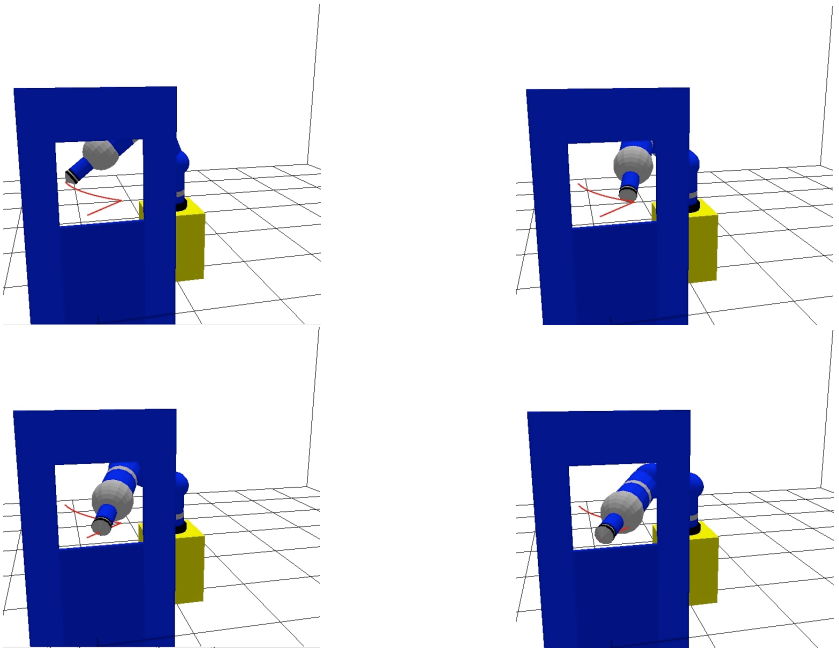


Planner	Mean Error	Max Error	# nodes
TCMP_KINO ($N = 10$)	0.0168 cm	0.0754 cm	91.7
RRT_LIKE ($N = 10$)	0.6649 cm	2.9790 cm	68.2
RRT_LIKE ($N = 100$)	0.1149 cm	1.5371 cm	1179.2

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

3-dof free-flying base has to follow a 33 m path with its end-point inside a pipe of variable radius. The averaged results have been obtained with a random choice of the vector $\tilde{w}$ and a number of path samples $N = 10$. The Eulero integration step used to generate motion with (5–6) has been set to $\Delta s = 0.01$. The mean and maximum errors were equal respectively to 0.0386 cm and 0.3223 cm, while the number of nodes in the tree was 99.6. The averaged running time was about 2 seconds.

Overall, our numerical results indicate that the TCMP_KINO outperforms the RRT_LIKE algorithm both in terms of tracking error and in terms of complexity. Even better (in principle, arbitrary) accuracy can be obtained by using a higher order integration algorithm without increasing the complexity of the tree. Hence, the running time of TCMP_KINO nicely scales with the complexity of the planning problem, independently of the accuracy required by the task tracking. Videos of the experiments are available at



Planner	Mean Error	Max Error	# nodes
TCMP_KINO ($N = 10$)	0.0098 cm	0.0436 cm	834.6
RRT_LIKE ($N = 100$)	0.1479 cm	0.8608 cm	1058.9

Figure 6: Second planning experiment on the DLR Light Weight Robot: comparison between TCMP_KINO with a number of path samples $N = 10$ and RRT_LIKE with $N = 100$.

`http://www.dis.uniroma1.it/labrob/research/TCMP.html`.

7 Conclusion

We have presented a randomized kinodynamic technique for solving the Task-Constrained Motion Planning problem in redundant robotic systems, possibly subject to nonholonomic 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 complexity of the roadmap. Preliminary results are very encouraging and confirm the performance improvement obtained with the proposed technique.

Future work will address several points, among which we mention:

- a careful study of the most convenient heuristics for choosing $\tilde{w}$ in the motion generation scheme;

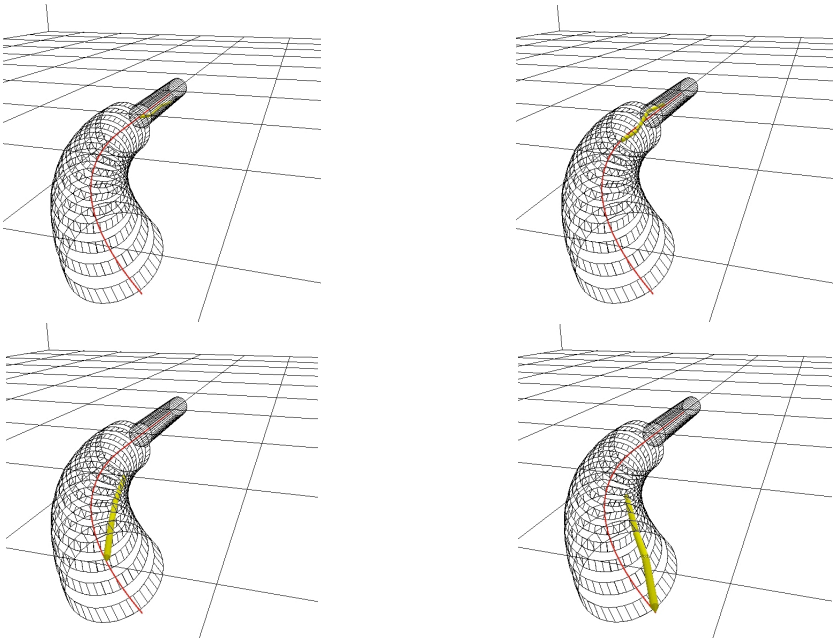


Figure 7: Third planning experiment: a 7-dof robot with free-flying base moving its end-point along an inspection path inside a variable radius pipe.

- an optimization of the collision checking procedure, currently performed on each configuration created by the motion generation scheme;
- the application of the planner to more complex robotic structures, such as humanoids.

Finally, in view of some preliminary analysis as well as of the obtained results, we conjecture that the probabilistic completeness of the RRT algorithm is preserved in our planning scheme, despite the fact that the construction of the tree takes place in the task constrained space $\mathcal{C}_{\text{task}}$. Future work will 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] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: Using task space to guide conguration space search," *2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1875–1880, 2005.

[8] 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.

[9] 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.

[10] 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.

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

[14] 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.

[15] 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.

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

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

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

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

[20] 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.