2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Planning Safe Cyclic Motions under Repetitive Task Constraints

Massimo Cefalo, Giuseppe Oriolo, Marilena Vendittelli

Abstract— **We consider motion planning in the presence of obstacles for redundant robotic systems subject to repetitive task constraints. For this open problem, we present a novel control-based randomized planner which produces cyclic, collision-free paths in configuration space and guarantees continuous satisfaction of the task constraints. In particular, the proposed algorithm relies on bidirectional search and loop closure in the task-constrained configuration space. Planning experiments on a simple 3R planar robot and the KUKA LWR-IV 7-dof manipulator are reported to show the effectiveness of the proposed method.**

I. Introduction

The operation of robotic systems invariably involves some form of task constraint. In the industrial context, typical tasks concern the manipulator 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). In service robotics, more general tasks can be assigned, ranging from tracking visual features to maintaining mechanical balance. A special case, which is particularly relevant in industrial applications but may also arise in other situations, is that of repetitive tasks; i.e., tasks that must be repeated over and over, and are therefore described by closed paths in the task space.

Redundant robots possess more degrees of freedom than those strictly necessary to accomplish a given task, and therefore offer increased dexterity and flexibility for pursuing additional objectives, among which one of the most important is avoidance of workspace obstacles. Traditionally, task-constrained motion in configuration space for redundant robots is generated through kinematic control techniques [1], [2], [3], [4]. These are on-line motion generation schemes that use a generalized inverse of the task Jacobian (e.g., the pseudoinverse), possibly with the addition of internal motions that do not perturb the execution of the task (null-space motions) and are therefore used for local optimization. However, researchers readily identified a critical issue of these schemes when used on repetitive tasks: in general, closed paths in the task space do not result in closed paths in the joint space. This is clearly a drawback, because it means that the robot motion is essentially unpredictable from cycle to cycle. Such behavior is particularly undesirable in human-robot interaction scenarios, because it ultimately hinders the legibility of the robot movements by the human.

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Rome, Italy. E-mail: {cefalo,oriolo,venditt}@dis.uniroma1.it. This work is supported by the EU FP7 ICT-287513 SAPHARI project.

With reference to the case of pure generalized inversion (no null-space motions), Shamir and Yomdin [5] identified a quite restrictive structural condition that the generalized inverse must satisfy in order to possess the *cyclicity* (also called *repeatability*) property. This condition, which was further refined in [6], was exploited to design cyclic generalized inverses, e.g., in [7] and to achieve asymptotically cyclic kinematic control in [8]. Other works on cyclicity include, e.g., [9], [10], [11]. However, even when a cyclic generalized inverse is used, there is no space left for additional objectives such as obstacle avoidance, because the addition of null-space motions would destroy the cyclicity property.

An exception to the above situation is represented by kinematic control schemes that rely on task augmentation to enforce a one-to-one mapping between task and configuration space. The archetype of this approach is the Extended Jacobian method [12], [13]. This technique is guaranteed to produce cyclic configuration paths in response to closed task paths, but only on the condition that algorithmic singularities are not encountered. Even neglecting for a moment this pitfall, however, it is impossible to guarantee that the solution paths are collision-free, essentially due to the fact that obstacle avoidance cannot be effectively encoded in an additional equality task.

An important observation is that repetitive tasks are usually assigned and known in advance. We argue, therefore, that off-line planning is the best approach to generate safe cyclic paths in configuration space when faced with this kind of tasks. In the literature, under the name (Task-)Constrained Motion Planning (for short, (T)CMP) one may find several methods for generating collision-free robot motions in the presence of (task) constraints; examples include [14], [15], [16], [17], [18], [19]. These techniques, however, do not consider the additional requirement that the paths in configuration space must be cyclic.

The solution approach proposed in this paper builds upon our previous work [17], in which a randomized planner was presented for solving the TCMP problem. In particular, a specialized motion generation scheme was introduced to guarantee continued satisfaction of the task constraint throughout the motion. Here we use the same basic scheme, but to guarantee that the generated paths in configuration space are cyclic, a bidirectional search is performed by growing two Rapidly-exploring Random Trees in the task-constrained configuration space: the first proceeds in the forward direction of the task space path, whereas the second moves backwards. When the two trees become sufficiently near, they are joined by a loop closure procedure that is designed using a feedback control technique.

The paper is organized as follows. Section II introduces the basic framework and formally defines the problem. The proposed solution method is described in Section III while the results of some planning experiments are presented in Section IV to illustrate its performance. Some future work is mentioned in the concluding section.

II. Problem Formulation

Since the problem addressed in this paper is a special case of Task-Constrained Motion Planning (TCMP), we will adopt the terminology and framework introduced in [17]. With respect to that work, which considers general robotic systems, here it is assumed for simplicity that the robot is not subject to nonholonomic constraints. This means, for example, that wheeled mobile manipulators are excluded from the developments to be presented. The extension to such systems is however relatively straightforward and is the subject of ongoing work.

Consider a robot whose configuration $\boldsymbol{q}$ takes value in an n_q-dimensional configuration space $\mathcal{C}$. The robot body moves in a workspace $\mathcal{W}$ (a subset of $I\!R^2$ or $I\!R^3$) that contains obstacles. Since no differential constraints act on the system, its kinematic model consists of simple integrators. For path or motion planning purposes, it is appropriate to use the geometric form of such model, which specifies the admissible tangent vectors to a configuration space path $\boldsymbol{q}(s)$, where s is a path parameter, as

$$\boldsymbol{q}' = \tilde{\boldsymbol{v}}, \tag{1}$$

with the notation $(\ )' = d(\ )/ds$. This model simply expresses the fact that the generalized coordinates $\boldsymbol{q}$ can move arbitrarily in $\mathcal{C}$. The tilde over the $\boldsymbol{v}$'s indicates that these are *geometric* inputs, rather than velocity inputs. If $s = t$, a trajectory is directly planned rather than a path.

Consider now a robot task expressed in coordinates $\boldsymbol{t}$, which take values in an n_t-dimensional task space $\mathcal{T}$. The $\boldsymbol{t}$ coordinates depend on the $\boldsymbol{q}$ coordinates through the kinematic map

$$\boldsymbol{t} = \boldsymbol{f}(\boldsymbol{q}). \tag{2}$$

At the differential level, it is

$$\boldsymbol{t}' = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{q}' = \boldsymbol{J}(\boldsymbol{q})\tilde{\boldsymbol{v}},$$

where $\boldsymbol{J}(\boldsymbol{q}) = d\boldsymbol{f}/d\boldsymbol{q}$ is the $n_t \times n$ task Jacobian which maps geometric inputs (configuration-space tangent vectors) to task-space tangent vectors.

We will assume that $n_q > n_t$, i.e., the robot is kinematically redundant with respect to the task of interest. This means that the inverse image $\bar{\boldsymbol{q}} = \boldsymbol{f}^{-1}(\bar{\boldsymbol{t}})$ in $\mathcal{C}$ of a point $\bar{\boldsymbol{t}}$ in $\mathcal{T}$ is not unique in general. In particular, this image may be *(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 [20]. Task points that map to the first group include *regular* and *coregular* points (the latter are also called avoidable singularities), whereas only *singular* points (unavoidable singularities) map to the second group.

Assume now that the assigned task is a *closed* path $\boldsymbol{t}_d(s)$, with $s \in [0,1]$ w.l.o.g. and $\boldsymbol{t}_d(0) = \boldsymbol{t}_d(1)$. For technical reasons, we will suppose that $\boldsymbol{t}_d(s)$ is differentiable and, for the problem to be well-posed, that:

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

where $\mathcal{T}^*$ is the *non-singular task space*, defined as the set of regular and coregular task points of $\mathcal{T}$. The initial joint configuration $\boldsymbol{q}_{\text{start}}$ is assigned[1], with $\boldsymbol{f}(\boldsymbol{q}_{\text{start}}) = \boldsymbol{t}_d(0)$.

The Cyclic Task-Constrained Motion Planning (C-TCMP) problem consists in finding a configuration space path $\boldsymbol{q}(s), s \in [0,1]$, such that:

1) $\boldsymbol{t}(s) = \boldsymbol{f}(\boldsymbol{q}(s)) = \boldsymbol{t}_d(s)$, for $s \in [0,1]$ (the desired path is executed in $\mathcal{T}$);
2) $\boldsymbol{q}(0) = \boldsymbol{q}(1)$ (the motion is cyclic in $\mathcal{C}$);
3) the robot does not collide with obstacles or with itself.

The planning space for the C-TCMP problem is

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

The manifold $\mathcal{C}_{\text{task}}$, called *task-constrained configuration space* in the following, is a natural *foliation*, with s as a parameter. A generic *leaf* is defined as

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

and we have

$$\mathcal{C}_{\text{task}} = \bigcup_{s \in [0,1]} \mathcal{L}(s).$$

Since the desired task path is closed, we have $\mathcal{L}(0) = \mathcal{L}(1)$.

The existence of a solution to the C-TCMP problem depends on the obstacle placement, and in particular by the connectivity of $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{task}}$, the portion of the free configuration space that is compatible with the task path constraint.

III. The C-TCMP planner

Our approach to the solution of the C-TCMP problem is illustrated in Fig. 1. In summary, a bidirectional search of the task-constrained configuration space $\mathcal{C}_{\text{task}}$ is performed by growing two Rapidly-exploring Random Trees (RRTs, see [21]). Both trees are rooted at $\boldsymbol{q}_{\text{start}}$, i.e., on $\mathcal{L}(0)$; but the first explores $\mathcal{C}_{\text{task}}$ in the direction of increasing s, whereas the second proceeds in the opposite direction. In view of the constraint represented by the desired task path, tree extension in our planner is obtained via a control-based variation [17] of the basic RRT mechanism. When the two trees become sufficiently near, they are joined by a loop closure procedure.

For building the trees, we sample the desired task path $\boldsymbol{t}_d(s)$ using a sequence $\{s_0 = 0, s_1, \ldots, s_{N-1}, s_N = 1\}$ of $N+1$ values of the path parameter s. Correspondingly, we use the shorthand notations $\boldsymbol{t}_{d,i} = \boldsymbol{t}_d(s_i)$ and $\mathcal{L}_i = \mathcal{L}(s_i)$. Recall that it is $\boldsymbol{t}_{d,0} = \boldsymbol{t}_{d,N}$ and $\mathcal{L}_0 = \mathcal{L}_N$.

[1]If needed, $\boldsymbol{q}_{\text{start}}$ can be preliminarily computed by inverse kinematics.

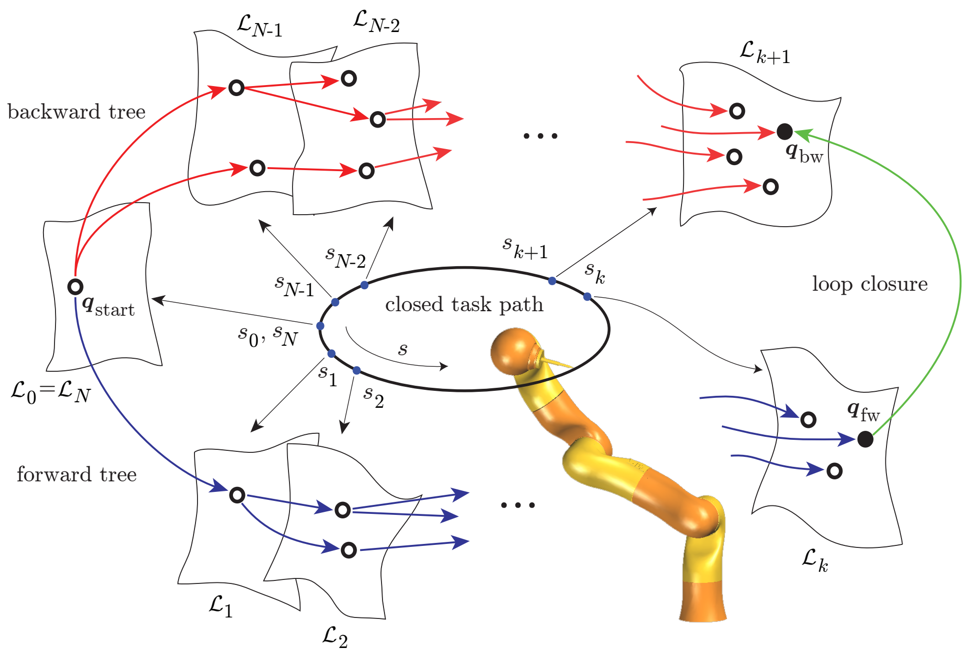

Fig. 1. The proposed C-TCMP planner relies on bidirectional search and loop closure in the task-constrained configuration space.

A. Motion Generation

The tree edges are subpaths in configuration space produced by using the following motion generation scheme:

$$\boldsymbol{q}' = \tilde{\boldsymbol{v}} \tag{3}$$
$$\tilde{\boldsymbol{v}} = \boldsymbol{J}^\dagger(\boldsymbol{q})(\boldsymbol{t}_d' + k_t\,\boldsymbol{e}_t) + (\boldsymbol{I} - \boldsymbol{J}^\dagger(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}))\tilde{\boldsymbol{w}}, \tag{4}$$

where $\boldsymbol{J}^\dagger$ is the pseudoinverse of the task Jacobian $\boldsymbol{J}$, k_t 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_q-vector that represents a *residual* input. One has $\boldsymbol{J}^\dagger = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}$ when $\boldsymbol{J}$ is full row rank; in our planner, this assumption is always satisfied because singular configurations are discarded (see Sect. III-B). Use of the above scheme guarantees $\boldsymbol{e}_t' = -k\,\boldsymbol{e}_t$, i.e., exponentially stable[2] tracking of the desired task path, regardless of the choice of $\tilde{\boldsymbol{w}}$.

For any choice of $\tilde{\boldsymbol{w}}(s)$, $s \in [0,1]$, and starting from a generic leaf of $\mathcal{C}_{\text{task}}$, integration of (3–4) provides a configuration path that starts from the current leaf and traverses subsequent leaves, always remaining in $\mathcal{C}_{\text{task}}$. A numeric solver can be used to actually perform the integration.

In principle, one could use a different parameterization for the task space and configuration paths [17]. This would allow, for example, to perform a self-motion at the configuration level so as 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.

[2]Since this is a planning scheme, stability is required for reducing the drift associated to a numerical integration of (3–4).

B. Tree Extension

The forward tree T_{fw} is rooted at $\boldsymbol{q}_{\text{start}}$. At each iteration, a random configuration $\boldsymbol{q}_{\text{rand}}$ is first generated[3] in $\mathcal{C}_{\text{task}}$; this is done by picking a value for $s \in \{s_1, \ldots, s_{N-1}\}$, and then choosing one of the inverse kinematic solutions for the corresponding sample of $\boldsymbol{t}_d(s)$. A scan of T_{fw} is then performed to identify its vertex (configuration) that is closest to $\boldsymbol{q}_{\text{rand}}$; call $\boldsymbol{q}_{\text{near}}$ this configuration and s_i the parameter value associated to the leaf $\mathcal{L}_i$ where $\boldsymbol{q}_{\text{near}}$ is located. At this point, a fixed number of different constant values are randomly generated for the residual vector $\tilde{\boldsymbol{w}}$, with the constraint that the norms of the two terms in the rhs of (4) are in a certain proportion. For each value of $\tilde{\boldsymbol{w}}$, the motion generation scheme (3–4) is then integrated numerically starting from $\boldsymbol{q}_{\text{near}}$ at $s = s_i$ and ending at $s = s_{i+1}$. Only the value of $\tilde{\boldsymbol{w}}$ that results in the final configuration closest to $\boldsymbol{q}_{\text{rand}}$ is retained. The corresponding subpath joins $\boldsymbol{q}_{\text{near}}$, belonging to $\mathcal{L}_i$, to a new configuration $\boldsymbol{q}_{\text{new}}$, that belongs to the next leaf $\mathcal{L}_{i+1}$.

Once the subpath has been generated, it is checked for collision with obstacles as well as for the occurrence of singularities. If the test is passed, $\boldsymbol{q}_{\text{new}}$ and the subpath from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ are added to T_{fw} as a vertex and an edge, respectively.

The extension mechanism of the backward tree T_{bw} is identical, with the only difference that the integration is performed backwards, i.e., over *decreasing* values of s. That is, edges of from T_{bw} starting from $\boldsymbol{q}_{\text{start}}$ on $\mathcal{L}_0 = \mathcal{L}_N$ will lead to vertices on $\mathcal{L}_{N-1}$, and so on (see Fig. 1). Recall

[3]Our experiments have shown that generating $\boldsymbol{q}_{\text{rand}}$ in $\mathcal{C}_{\text{task}}$, although more expensive in itself, improves considerably the overall performance of the planner with respect to simply taking $\boldsymbol{q}_{\text{rand}}$ in $\mathcal{C}$.

that system (3–4) is symmetric, and therefore any backward motion can be reversed to the direction where s increases.

We emphasize that, unlike sampling-based constrained motion planners, the use of the motion generation scheme (3–4) guarantees that the task variables move along the desired path throughout the motion. In particular, the tracking accuracy can be arbitrarily increased by reducing the stepsize for integrating (3–4), without any consequence on the size of the tree, to which only $\boldsymbol{q}_{\rm new}$ will be added in any case.

The planning algorithm proceeds by alternately growing $T_{\rm fw}$ and $T_{\rm bw}$. As customary in bidirectional search, extension steps are occasionally replaced (a parameter controls this event) by connection steps aimed at reducing the gap between the two trees. Whenever a pair of vertices is generated that belong to different trees and lie on two adjacent leaves, the algorithm invokes the loop closure procedure.

C. Loop Closure

As shown in Fig. 1, once two vertices (configurations) $\boldsymbol{q}_{\rm fw} \in \mathcal{L}_k$ and $\boldsymbol{q}_{\rm bw} \in \mathcal{L}_{k+1}$ have been generated by $T_{\rm fw}$ and $T_{\rm bw}$, respectively, they must be joined with a collision-free path contained in $\mathcal{C}_{\rm task}$ in order to 'close the loop' and produce a cyclic configuration space path that solves the C-TCMP problem. In this phase, in place of (3–4) we use a different motion generation scheme that is inspired to the self-motion stabilization scheme proposed in [22].

The basic idea here is to identify a suitable subset of $n_q - n_t$ 'redundant' configuration variables, and to perform the desired reconfiguration on these variables. The remaining 'base' variables will move so as to ensure that the desired task space subpath from $\boldsymbol{t}_{d,k}$ to $\boldsymbol{t}_{d,k+1}$ is followed. Under certain conditions, this guarantees that the base variables will also converge to their desired value. This can be formalized as follows.

Partition the configuration vector $\boldsymbol{q}$ as $(\boldsymbol{q}^{\rm red}, \boldsymbol{q}^{\rm base})$, where $\boldsymbol{q}^{\rm red}$ is $(n_q - n_t)$-dimensional and $\boldsymbol{q}_{\rm base}$ is n_t-dimensional. Correspondingly, the task Jacobian is also partitioned in an $n_t \times (n_q - n_t)$ submatrix $\boldsymbol{J}_{\rm red} = \partial \boldsymbol{f}/\partial \boldsymbol{q}^{\rm red}$ and an $n_t \times n_t$ submatrix $\boldsymbol{J}_{\rm base} = \partial \boldsymbol{f}/\partial \boldsymbol{q}^{\rm base}$. Assume that the partition has been made in such a way that $\boldsymbol{J}_{\rm base}$ is nonsingular at $\boldsymbol{q}_{\rm fw}$. Reconfiguration from $\boldsymbol{q}_{\rm fw}$ to $\boldsymbol{q}_{\rm bw}$ in $\mathcal{C}_{\rm task}$ can then be obtained by letting

$$(\boldsymbol{q}^{\rm red})' = k_{\rm red}(\boldsymbol{q}^{\rm red}_{\rm bw} - \boldsymbol{q}^{\rm red}) \quad (5)$$
$$(\boldsymbol{q}^{\rm base})' = \boldsymbol{J}^{-1}_{\rm base}(\boldsymbol{q})(\boldsymbol{t}'_d + k_t\boldsymbol{e}_t - \boldsymbol{J}_{\rm red}(\boldsymbol{q})(\boldsymbol{q}^{\rm red})') \quad (6)$$

where $k_{\rm red}$ is a positive constant, $\boldsymbol{q}^{\rm red}_{\rm bw}$ is the value of $\boldsymbol{q}^{\rm red}$ at $\boldsymbol{q}_{\rm bw}$, and $\boldsymbol{t}'_d$ indicates the geometric derivative of the desired task path from $\boldsymbol{t}_{d,k}$ to $\boldsymbol{t}_{d,k+1}$.

By integrating eqs. (5–6) from $\boldsymbol{q}_{\rm fw}$ at $s = s_k$, the $\boldsymbol{q}^{\rm red}$ variables will converge exponentially to their destination $\boldsymbol{q}^{\rm red}_{\rm bw}$, while the final value of the $\boldsymbol{q}^{\rm base}$ variables will belong to the finite set of the inverse kinematic solutions corresponding to $\boldsymbol{t}_{d,k+1}$. In particular, it is easy to prove that $\boldsymbol{q}^{\rm base}$ will converge exactly to $\boldsymbol{q}^{\rm base}_{\rm bw}$ provided that *(i)* $\boldsymbol{q}^{\rm base}_{\rm fw}$ and $\boldsymbol{q}^{\rm base}_{\rm bw}$ belong to the same class of kinematic solutions (e.g., elbow-up or elbow down), and *(ii)* $\boldsymbol{J}_{\rm base}$ remains nonsingular throughout the motion. Under these conditions, therefore, the configuration variables will move from $\boldsymbol{q}_{\rm fw}$ to $\boldsymbol{q}_{\rm bw}$ while satisfying the task constraint, and hence the loop will be effectively closed. If $\boldsymbol{J}_{\rm base}$ becomes singular (or better, approaches a singularity) during the use of (5–6), it is sufficient to change the partition of $\boldsymbol{q}$ and re-apply loop closure from the current configuration.

In practice, the above conceptual scheme is used in our planner without changing the partition of $\boldsymbol{q}$. If $\boldsymbol{J}_{\rm base}$ becomes singular, or if a collision is found, we simply abort the loop closure phase and go back to the tree extension phase, because the probabilistic completeness property of RRT guarantees that sooner or later another pair $(\boldsymbol{q}_{\rm fw}, \boldsymbol{q}_{\rm bw})$ will be generated on which loop closure will be successful.

When closure is obtained, a solution to the C-TCMP problem is reconstructed by patching together the subpath from $\boldsymbol{q}_{\rm start}$ to $\boldsymbol{q}_{\rm fw}$ on $T_{\rm fw}$, the loop closure subpath from $\boldsymbol{q}_{\rm fw}$ to $\boldsymbol{q}_{\rm bw}$, and the subpath from $\boldsymbol{q}_{\rm bw}$ to $\boldsymbol{q}_{\rm start}$ on $T_{\rm bw}$ (in the reverse direction).

IV. Planning Experiments

The proposed C-TCMP planner has been implemented in Kite, a cross-platform motion planning software produced by Kineo CAM, on a 64-bit Intel Core i5-2320 CPU running at 3 GHz. We report planning results for scenarios of increasing difficulty. The first involves a simple 3R planar manipulator, while the second and the third refer to the KUKA LWR-IV 7-DOF robot. Animations of the generated motions are contained in the video clip accompanying this submission and are available at http://www.dis.uniroma1.it/~labrob/research/C-TCMP.html.

In the first scenario, a 3R planar manipulator must move its tip along an elliptical path while avoiding three small obstacles. In this case, we have a single degree of redundancy. We used[4] $N + 1 = 11$ equispaced samples of the desired task path and generated motions using Euler's method with integration step $\Delta s = 0.002$. The gain k_t was set to 100. The norm of the null space vector in eq. (4) is constrained to be at most 150% of that of the first term. Figure 2 shows a cyclic solution which was computed by our planner in about 4 s (the average running time computed over ten runs was around 5 s). The forward and backward trees contain at the end 24 and 11 vertices, respectively. The mean and the maximum value of the task error norm over the whole path are, respectively, 0.0729 mm and 0.1354 mm, confirming that the proposed method guarantees continued satisfaction of the task constraint. Cyclicity in configuration space is confirmed by Fig. 5, which shows that the displacement with respect to the initial configuration returns to zero at the end of the motion. For comparison, the accompanying video shows the path produced by simple pseudoinverse control, which is not cyclic — as expected — and also leads to collisions with the obstacles.

In the second scenario, the KUKA LWR-IV must draw a circle on a whiteboard while avoiding collisions with the table on which it is mounted, the wall, the whiteboard itself

[4]See [17] for a discussion on the choice of N.

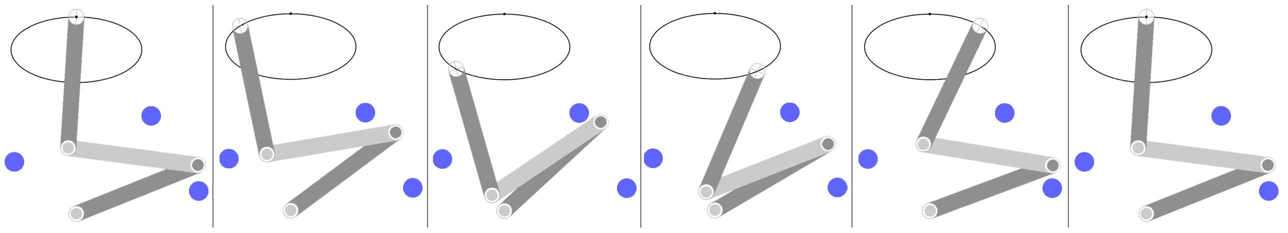

Fig. 2. Planning experiment on a 3R planar robot: Samples from the solution. The first and the last sample confirm that cyclicity has been achieved.

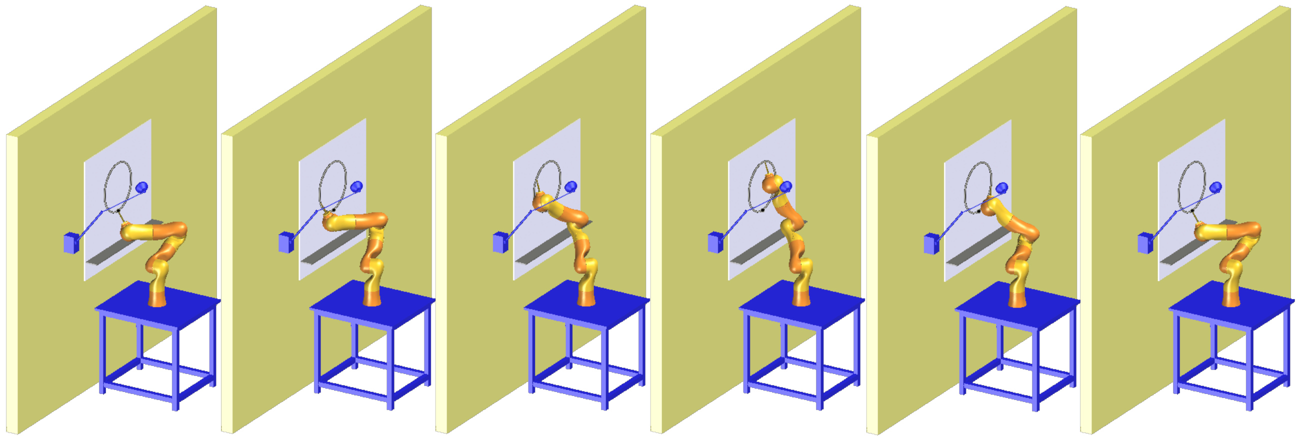

Fig. 3. First planning experiment on the LWR-IV: Samples from the solution. The first and the last sample confirm that cyclicity has been achieved.

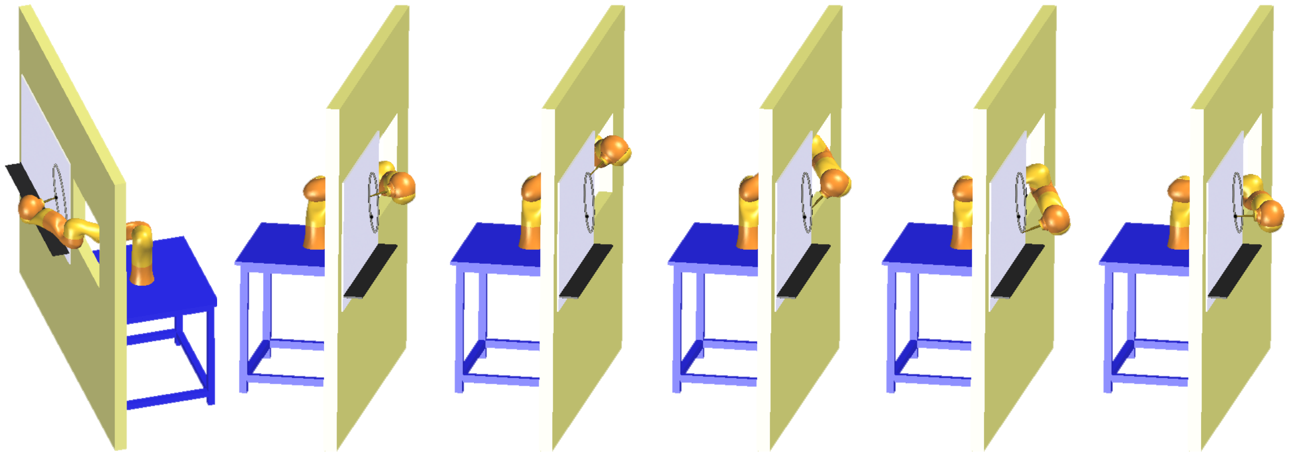

Fig. 4. Second planning experiment on the LWR-IV: Samples from the solution. The first and the last sample confirm that cyclicity has been achieved.

(simple contact with the pen is obviously allowed) and an articulated lamp protruding from the wall. Self-collisions are also avoided. Since the drawing task is 3-dimensional, the degree of redundancy is $6 - 3 = 3$ (the wrist roll is frozen because it is irrelevant for the task). The planner parameters are the same of the previous case. The solution shown in Fig. 3 took 83 s to compute, with a final size of the forward and backward trees of 116 and 33 vertices, respectively. The task error is again negligible (mean value 0.076 mm and maximum value 0.1814 mm). Figure 6 shows that cyclicity has been achieved also in this case.

The third scenario refers to a similar task (draw an ellipse on the whiteboard) but the LWR-IV is now placed on the opposite side of the wall and must maneuver through a small window to reach the whiteboard. Using once again the same parameters of the previous experiments, the solution of Figs. 4 and 7 was obtained in 32 s (consider that the size of $\mathcal{C}_{\text{task}} \cap \mathcal{C}_{\text{free}}$ is significantly smaller than in the second scenario). At the end, the forward and backward trees contain 29 and 34 vertices, respectively, while the mean and the maximum task errors are 0.0275 mm and 0.0418 mm.

V. Conclusions

For redundant robotic systems subject to repetitive task constraints, we have presented a control-based approach for planning cyclic, collision-free configuration space paths. We

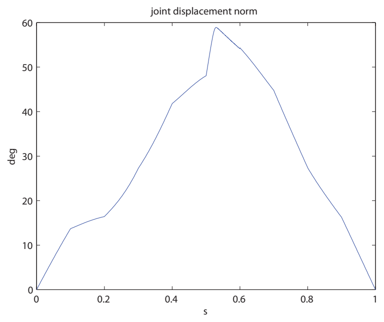

Fig. 5. Planning experiment on a 3R planar robot: Norm of $\boldsymbol{q} - \boldsymbol{q}_{\rm start}$.

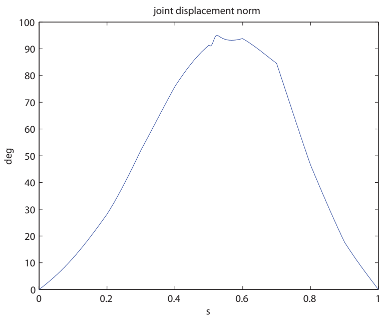

Fig. 6. First planning experiment on the LWR-IV: Norm of $\boldsymbol{q} - \boldsymbol{q}_{\rm start}$.

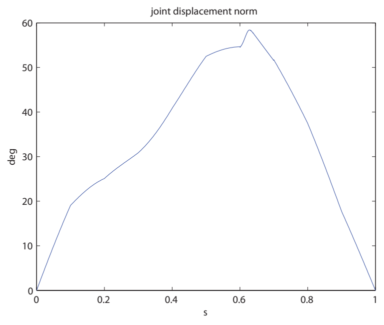

Fig. 7. Second planning experiment on the LWR-IV: Norm of $\boldsymbol{q} - \boldsymbol{q}_{\rm start}$.

believe that our contribution fills a void in the literature. In fact, on the one hand kinematic control schemes are either non-cyclic or, when they are cyclic, they leave no space for guaranteed obstacle avoidance. On the other hand, existing task-constrained motion planners cannot produce cyclic paths in configuration space.

Our randomized C-TCMP planner relies on bidirectional search and loop closure in the task-constrained configuration space. As a consequence, it produces cyclic paths on which continued satisfaction of the task constraint is guaranteed. Planning experiments on a simple 3R planar manipulator and a KUKA LWR-IV robot have been presented to show the effectiveness of the proposed approach.

Future work will be aimed at several objectives, such as:

- the use of specific metrics for $\mathcal{C}_{\rm task}$;
- a formal proof of probabilistic completeness for the planner;
- the extension of the proposed approach to robotic systems subject to nonholonomic constraints (e.g., wheeled mobile manipulators).

REFERENCES

[1] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[2] C. A. Klein and C. H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 13, no. 3, pp. 245–250, 1983.

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

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

[5] T. Shamir and Y. Yomdin, "Repeatability of redundant manipulators: mathematical solution of the problem," *IEEE Trans. on Automatic Control*, vol. 33, no. 11, pp. 1004–1009, 1988.

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

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

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

[9] R. Mukherjee, "Design of holonomic loops for repeatability in redundant manipulators," in *1995 IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995, pp. 2785–2790.

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

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

[12] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *1985 IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, 1985, pp. 722–728.

[13] P. H. Chang, "A closed-form solution for inverse kinematics of robot manipulators with redundancy," *IEEE Trans. on Robotics and Automation*, vol. 3, no. 5, pp. 393–403, 1987.

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

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

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

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

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

[19] C. Suh, B. Kim, and F. Park, "The tangent bundle RRT algorithms for constrained motion planning," in *13th World Congress in Mechanism and Machine Science*, Guanajuato, Mexico, 2011, pp. 1–5.

[20] J. W. Burdick, "On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds," in *1989 IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 264–270.

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

[22] G. Oriolo, "Stabilization of self-motions in redundant robots," in *1994 IEEE Int. Conf. on Robotics and Automation*, San Diego, CA, 1994, pp. 704–710.