2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Dynamically Feasible Task-Constrained Motion Planning with Moving Obstacles

Massimo Cefalo, Giuseppe Oriolo

Abstract— **We present a randomized algorithm for planning dynamically feasible motions of robots subject to geometric task constraints in the presence of moving obstacles. The proposed method builds upon our previous results on task-constrained motion planning with moving obstacles. With respect to our previous formulation, the inclusion of bounds on the available actuator torques leads to the adoption of an acceleration-level motion generation scheme. Therefore, the new planner must operate in a task-constrained state space extended with time. The generated trajectories are collision-free, obey velocity and torque bounds, and satisfy the task constraint with arbitrary accuracy. The effectiveness of the proposed approach is shown by results on various scenarios involving a 7-dof manipulator.**

I. Introduction

Basic motion planning considers a robot moving among fixed obstacles. The extension to the case of moving obstacles is relevant in practice, due to the increasing shift from industrial to service applications of robotics. However, even for known obstacle motions, this extension proves to be very challenging, starting already with the most elementary case of a single-body robot with unbounded velocity [1].

Early attempts to solve motion planning problems with moving obstacles were based on extensions of combinatorial or sampling-based methods, see, e.g., [2], [3], [4]. A common device was the extension of the planning space from the simple configuration (or state) space to a configuration-time (state-time) space. Another approach was based on the so-called velocity obstacle technique, see [5], [6], [7]. All these methods, however, cannot be used in high-dimensional configuration spaces due to their computational complexity.

A fallout of considering moving obstacles is that the available actuator torques must be taken into account. In fact, even in kinematically feasible plan, some collision avoidance maneuvers may prove beyond the dynamic possibilities of the robot. Neither does slowing down the plan work in general, because in the presence of moving obstacles collisions may appear. In [8], where the expression *kinodynamic planning* was coined, the planning problem was solved for a point mass robot subject to acceleration constraints. The sampling-based algorithm proposed in [9] for the case of moving obstacles considers both kinematic and dynamic constraints.

None of the above methods allows task constraints, which however arise in many practical applications. For instance, manipulators used in industrial processes are frequently required to follow specific end-effector paths or trajectories for welding, drawing, cutting or assembling. Another example is a service robot that must maintain visual tracking of a target, or carry an object (such as a tray) with a certain orientation. Task-Constrained Motion Planning (henceforth TCMP) has therefore been recently acknowledged as an important extension of the basic motion planning problem.

Since task constraints restrict the set of feasible configurations to a lower-dimensional submanifold, random sampling of the configuration space is not effective. To solve this problem, a common approach is to use a standard randomized search algorithm (such as PRM [10] or RRT [11]) and then project the samples on the admissible submanifold; projection can be realized via randomized gradient descent, tangent space sampling or retraction, e.g., see [12]. A method in this class that allows torque limits is [13]. A different approach was presented in [14], where we introduced a control-based method for TCMP which avoids altogether the need for projection. The proposed algorithm guarantees continued satisfaction of the task constraints with a precision that can be arbitrarily improved without increase in complexity.

These previous works on TCMP deal only with fixed obstacles. In [15] we proposed an extension of our control-based planner to the case of moving obstacles. The method is purely kinematic and may handle velocity lints. Here, we develop that approach to account for the presence of bounds on the available actuator torques. Although this may seem as a relatively direct extension, it actually requires some radical changes. In particular, in the proposed approach the motion generation scheme is translated to the acceleration level, and the robot dynamic model is used within the planning phase. Accordingly, the search space is obtained by adding the time dimension to the state (as opposed to configuration) space.

Throughout the paper, the obstacles are assumed to move along trajectories that are known in advance. While this is certainly a simplification, there are plenty of applications in which such an assumption is acceptable, ranging from industrial workcells to digital animation. In any case, our investigation may be seen as a first step towards the solution of more challenging problems characterized by a reduced level of predictability of the obstacle motion.

The paper is organized as follows. In Section II we formulate the problem. Section III discusses the geometric structure of the search space. The proposed algorithm is described in Section IV, and validated in Section V. Possible future developments are hinted at in the concluding section.

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}@diag.uniroma1.it. This work is supported by the EU FP7 ICT-287513 SAPHARI project.

II. Problem Formulation

The robot moves in a workspace $\mathcal{W}$ (a subset of $I\!R^3$) that contains moving obstacles. Denote by $\boldsymbol{q}$ the n_q-dimensional configuration vector and by $\mathcal{C}$ the configuration space. Let $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$ be the volume occupied by the robot at configuration $\boldsymbol{q}$, and $\mathcal{O}(t) \subset \mathcal{W}$ be the obstacle region at time t. Throughout the paper, it is assumed that $\mathcal{O}(t) \subset \mathcal{W}$ is known for all t; i.e., the obstacle motion is fully predictable.

The robot dynamics is expressed in the Lagrangian form:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau},$$

where $\boldsymbol{B}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ collects velocity and gravitational terms, and $\boldsymbol{\tau}$ are the generalized forces (simply called torques below) provided by the actuators. We assume that the robot is subject to joint range limits, to maximum bounds $|\dot{\boldsymbol{q}}| \leq \boldsymbol{q}_M$ on the generalized velocities, and to maximum bounds $|\boldsymbol{\tau}| \leq \boldsymbol{\tau}_M$ on the torques. A trajectory in $\mathcal{C}$ satisfying all these bounds is called *feasible*.

The robot task is described by an n_y-dimensional vector $\boldsymbol{y}$, which takes values in the task space $\mathcal{Y}$. The value of $\boldsymbol{y}$ is related to that of $\boldsymbol{q}$ by a forward kinematic map $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q})$, whose differential version is $\dot{\boldsymbol{y}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$, with $\boldsymbol{J} = \partial \boldsymbol{f}/\partial \boldsymbol{q}$ the $n_y \times n_q$ task Jacobian. We assume $n_q > n_y$, i.e., the robot is redundant with respect to task $\boldsymbol{y}$.

Suppose now that a desired *path* $\boldsymbol{y}_d(s) \in \mathcal{Y}$ is assigned to the task coordinates, where $s \in [s_i, s_f]$ is a path parameter. The considered problem (Dynamically Feasible Task-Constrained Motion Planning with Moving Obstacles, or DF_TCMP_MO) consists in finding a *feasible trajectory* in configuration space such that the assigned task path is executed and collisions with obstacles are avoided. In our approach, such trajectory is actually the result of two components, i.e., a configuration space path and a time history along the path. In formal terms:

In formal terms, a solution to the DF_TCMP_MO problem consists of a path $\boldsymbol{q}(s) \in \mathcal{C}$, $s \in [s_i, s_f]$, and a continuous time history $s(t) : [0, T] \mapsto [s_i, s_f]$, such that :

1. $s(0) = s_i$ and $s(T) = s_f$;
   % task starts at $\boldsymbol{y}_d(s_i)$ and ends at $\boldsymbol{y}_d(s_f)$;
2. for all $t \in [0, T]$, it is $\boldsymbol{y}(t) = \boldsymbol{f}(\boldsymbol{q}(s(t))) = \boldsymbol{y}_d(s(t))$;
   % task is always on the assigned path
3. $\boldsymbol{q}(0) = \boldsymbol{q}_i$, $\dot{\boldsymbol{q}}(0) = \dot{\boldsymbol{q}}_i$, $\dot{\boldsymbol{q}}(T) = \dot{\boldsymbol{q}}_f$
   % assigned initial configuration is matched
   % assigned initial and final velocities are matched
4. for all $t \in [0, T]$, it is $|\dot{\boldsymbol{q}}(t)| \leq \dot{\boldsymbol{q}}_M$ and $|\boldsymbol{\tau}(t)| \leq \boldsymbol{\tau}_M$;
   % velocity and torque bounds are satisfied
5. for all $t \in [0, T]$, it is $\mathcal{R}(\boldsymbol{q}(s(t))) \cap \mathcal{O}(t) = \emptyset$;
   % collisions with moving obstacles are avoided

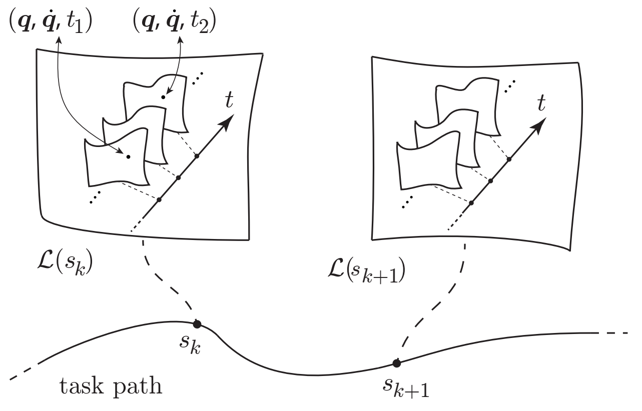
The generalized velocities and accelerations associated to a particular solution, which are both involved in condition 4, are easily computed as

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{q}'(s)\dot{s}(t)$$

and

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{q}''(s)\dot{s}^2(t) + \boldsymbol{q}'(s)\ddot{s}(t), \tag{1}$$

having used the notation $()' = d()/ds$.



$(\boldsymbol{q}, \dot{\boldsymbol{q}}, t_1)$
$(\boldsymbol{q}, \dot{\boldsymbol{q}}, t_2)$
t
t
$\mathcal{L}(s_k)$
$\mathcal{L}(s_{k+1})$
s_k
task path
s_{k+1}

Fig. 1. $\mathcal{S}_{\text{task}}$ is a foliation: each leaf $\mathcal{L}(s)$ is the set of points $(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \in \mathcal{S}$ such that $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are consistent with the task path constraint for a certain value of s, while t may assume any value.

Note that while the initial configuration $\boldsymbol{q}_i$ is given or preliminarily computed by inverse kinematics, the final configuration $\boldsymbol{q}(s_f) = \boldsymbol{q}(T)$ and the duration T of the motion are not assigned, and will be generated by the planner.

We emphasize that the time history $s(t)$ is not required to be monotonic: over time, s may increase or decrease, giving respectively a *forward motion* or a *backward motion* along the assigned task path. This possibility can be exploited to avoid moving obstacles while complying with the task.

III. Search Space

Define the robot state space as $\mathcal{X} = \mathcal{C} \times T_{\boldsymbol{q}}\mathcal{C}$, where $T_{\boldsymbol{q}}\mathcal{C}$ denotes the tangent space of $\mathcal{C}$ at $\boldsymbol{q}$. Due to the presence of moving obstacles, a robot state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ may be in collision at a certain time instant and free at another. Therefore, we must include time into the picture. This leads to defining the *state-time space* (STS for brevity) as $\mathcal{S} = \mathcal{X} \times [0, \infty)$, the *occupied STS* as $\mathcal{S}_{\text{occ}} = \{(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \in \mathcal{S} : \mathcal{R}(\boldsymbol{q}(t)) \cap \mathcal{O}(t) \neq \emptyset\}$, and the *free STS* as $\mathcal{S}_{\text{free}} = \mathcal{S} \setminus \mathcal{S}_{\text{occ}}$.

Similarly to moving obstacles, also the task path constraint reduces the admissible region of STS. In particular, define the *task-constrained STS* as the set of points of the state-time space whose state (generalized coordinates and velocities) is consistent with the assigned task path. In formula:

$$\begin{aligned}\mathcal{S}_{\text{task}} \;=\; & \{(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \in \mathcal{S} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{y}_d(s), \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{y}'_d(s)\dot{s}, \\ & \text{for some } s \in [s_i, s_f], \dot{s} \in (-\infty, \infty)\}.\end{aligned}$$

In $\mathcal{S}_{\text{task}}$, t is immaterial; i.e., for any point in $\mathcal{S}_{\text{task}}$, there exist infinite other points with the same state and different time instant $t \in [0, \infty)$. From a geometric viewpoint, $\mathcal{S}_{\text{task}}$ is a manifold with boundary which foliates:

$$\mathcal{S}_{\text{task}} = \cup_{s \in [s_i, s_f]} \mathcal{L}(s)$$

with each *leaf* $\mathcal{L}(s)$ associated to a value of $s \in [s_i, s_f]$:

$$\mathcal{L}(s) = \{(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \in \mathcal{S} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{y}_d(s), \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{y}'_d(s)\dot{s}\}.$$

Figure 1 illustrates the structure of $\mathcal{S}_{\text{task}}$.

The existence of a solution to the planning problem depends on the interplay between the task path and the obstacles' motion, and in particular on the connectedness of the search space $\mathcal{S}_{\text{task}} \cap \mathcal{S}_{\text{free}}$. However, even a candidate solution contained in such space may still turn out to be unfeasible when velocity and torque limits are considered.

IV. Proposed Planner

Our planner builds a tree in the search space $\mathcal{S}_{\text{task}} \cap \mathcal{S}_{\text{free}}$. The search is biased by N samples of the assigned task path, denoted by $\boldsymbol{y}_k = \boldsymbol{y}_d(s_k)$, corresponding to a predefined sequence $\{s_1 = s_i, \ldots, s_k, \ldots, s_N = s_f\}$ of values of s. Let $\mathcal{L}_k = \mathcal{L}(s_k)$ be the leaf associated to $\boldsymbol{y}_k$ (see Figure 1).

The root of the tree is the triplet $(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, 0)$, consisting of the initial robot state and time instant. This will be the only vertex on $\mathcal{L}_1$. All the other vertexes lie on some $\mathcal{L}_k$, $k = 2, \ldots, N$; in principle, there will be several vertexes on each leaf. Each vertex is a triplet $(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ representing a robot state and the time at which it was attained. An edge is a feasible subtrajectory joining two vertexes lying on adjacent leaves, produced by a suitable motion generation scheme.

A. Motion Generation

At the core of our proposed planner is a motion generation scheme that, starting from a generic vertex of the tree located on a certain leaf, produces a feasible subtrajectory that is contained in $\mathcal{S}_{\text{task}} \cap \mathcal{S}_{\text{free}}$ and lands on either the next or the previous leaf. The state and time instant at which landing occurs generate a new vertex. Due to the presence of torque bounds, such scheme must operate at the acceleration level.

Consider a generic vertex $V = (\boldsymbol{q}_V, \dot{\boldsymbol{q}}_V, t_V)$ on leaf $\mathcal{L}_k$. All vertexes on $\mathcal{L}_k$ share the same value of $s = s_k$, whereas the value of $\dot{s} = \dot{s}_V$ is different for each V, as a byproduct of the subtrajectory that generated that vertex. In view of (1), any $\ddot{\boldsymbol{q}}$ may be generated by choosing separately $\boldsymbol{q}''$ (the geometric acceleration) and $\ddot{s}$ (the rate of change of $\dot{s}$).

In particular, we choose $\ddot{s}$ as

$$\ddot{s} = \ddot{s}_V, \tag{2}$$

with $\ddot{s}_V$ a constant value chosen within a predefined range $[-c_{\max}, c_{\max}]$. As a consequence, the profile of $s(t)$ from t_V on will be quadratic. In particular, depending on the value of $\dot{s}_V$ and the chosen $\ddot{s}_V$, we may obtain essentially four kinds of motions of s over t, and correspondingly of $\boldsymbol{y}(s)$ over $\boldsymbol{y}_d(s)$: (1) a monotonic forward motion from s_k to s_{k+1} (2) a motion which moves initially backward from s_k but then reverses its direction before s_{k-1} and proceeds forward to reach s_{k+1} (3) a monotonic backward motion from s_k to s_{k-1} (4) a motion which moves initially forward from s_k but then reverses its direction before s_{k+1} and proceeds backwards to reach s_{k-1}.

The geometric acceleration is chosen as

$$\boldsymbol{q}''_V(s) = \boldsymbol{J}^{\dagger}(\boldsymbol{y}''_d - \boldsymbol{J}'\boldsymbol{q}' + \boldsymbol{K}_p\boldsymbol{e}_y + \boldsymbol{K}_d\boldsymbol{e}'_y) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})\boldsymbol{a}_V \tag{3}$$

where $\boldsymbol{J}^{\dagger}$ is the pseudoinverse of the task Jacobian, $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are positive definite gain matrices, $\boldsymbol{e}_y = \boldsymbol{y}_d - \boldsymbol{y}$ is the task error, $\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ is the orthogonal projection matrix in the null space of $\boldsymbol{J}$, and $\boldsymbol{a}_V$ is an arbitrary n_q-dimensional vector which produces internal motions without effect on the task. Note that $\boldsymbol{e}'_y = \pm\boldsymbol{y}'_d - \boldsymbol{J}\boldsymbol{q}'$, where the $+$ $(-)$ sign must be used in correspondence of increasing (decreasing) values of s, i.e., during a forward (backward) motion phase.

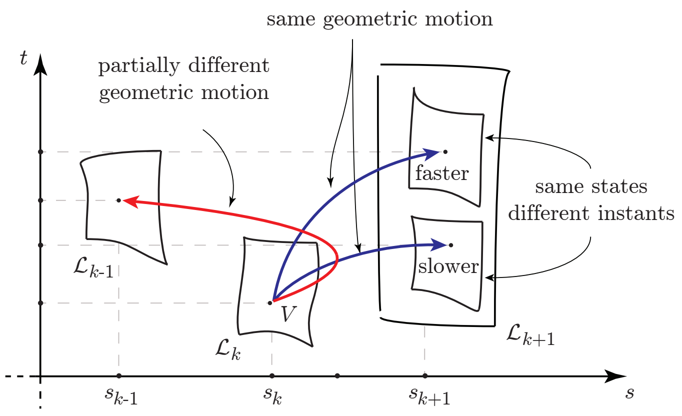


Fig. 2. An illustration of motion generation within our planner. In this particular case, it is assumed that $\dot{s}_k > 0$. The two monotonic forward motions (blue) correspond to the same choice of $\boldsymbol{a}_V$ (hence, of $\boldsymbol{q}''_V$) but different positive values of $\ddot{s}_V$. The non-monotonic motion (red) is generated by the same $\boldsymbol{a}_V$ but now with a negative $\ddot{s}_V$. The geometric motion is the same of the forward case until the direction is reversed.

Motion is then generated by integrating (2–3) from vertex V. In doing so, velocity and torque limits are continuously verified, together with avoidance of moving obstacles. If either of these is violated, motion generation is prematurely terminated. Otherwise, integration stops when the subtrajectory lands on an adjacent leaf to $\mathcal{L}_k$, be it $\mathcal{L}_{k+1}$ or $\mathcal{L}_{k-1}$.

Figure 2 illustrates some typical situations encountered when applying motion generation from a vertex on $\mathcal{L}_k$.

B. Tree Expansion

The planning tree is expanded using an RRT-like mechanism. At each iteration, a random task sample $\boldsymbol{y}_{\text{rand}} = \boldsymbol{y}_k$, with $k \in \{1, \ldots, N\}$, is extracted from the predefined sequence, and an inverse solution $\boldsymbol{q}_{\text{rand}} = \boldsymbol{f}^{-1}(\boldsymbol{y}_{\text{rand}})$ is computed. A random task-consistent generalized velocity $\dot{\boldsymbol{q}}_{\text{rand}} \in [-\dot{\boldsymbol{q}}_M, \dot{\boldsymbol{q}}_M]$ is chosen and attached to $\boldsymbol{q}_{\text{rand}}$. Finally, a time instant t_{rand} is sampled from $[0, t_{\max}]$, with $t_{\max}$ the largest time instant associated to a vertex in the current tree. By construction, $(\boldsymbol{q}_{\text{rand}}, \dot{\boldsymbol{q}}_{\text{rand}}, t_{\text{rand}})$ is a sample of $\mathcal{S}_{\text{task}}$.

At this point, the tree is searched for the closest vertex to $(\boldsymbol{q}_{\text{rand}}, \dot{\boldsymbol{q}}_{\text{rand}}, t_{\text{rand}})$, according to a suitably defined metric[1] Denote this vertex by $(\boldsymbol{q}_{\text{near}}, \dot{\boldsymbol{q}}_{\text{near}}, t_{\text{near}})$, and say it is located on a generic leaf $\mathcal{L}_k$.

Then, the tree is expanded from $V = (\boldsymbol{q}_{\text{near}}, \dot{\boldsymbol{q}}_{\text{near}}, t_{\text{near}})$ using the previous motion generation scheme with a randomly generated acceleration vector $\boldsymbol{a}_V$. As explained before, as soon as one of the two adjacent leaves $\mathcal{L}_{k+1}$ or $\mathcal{L}_{k-1}$ is reached by a feasible, collision-free subtrajectory, a new vertex is placed at the landing point. As a byproduct of the integration procedure, we obtain the time instant associated to the new vertex. Whenever a subtrajectory is discarded due to constraint violation, a new tree expansion takes place.

[1]In particular, a weighted sum is used to characterize distances in the state-time space $\mathcal{S}$.

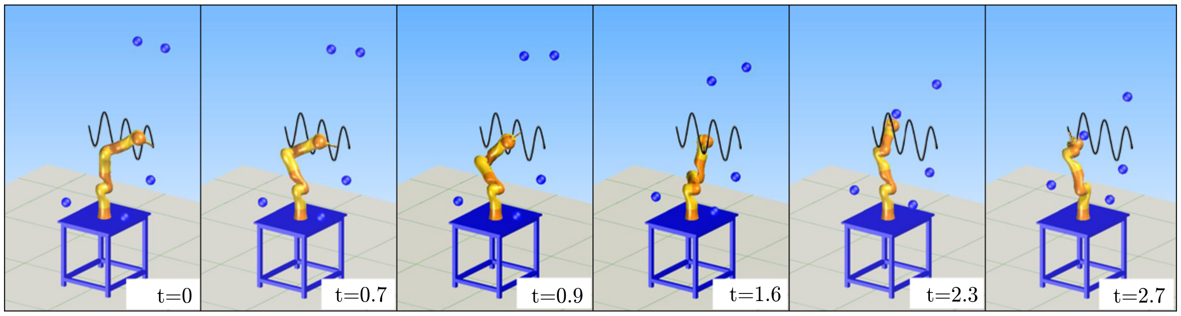


Fig. 3. Dynamically feasible planning experiment on the first scenario: sample frames from the solution.

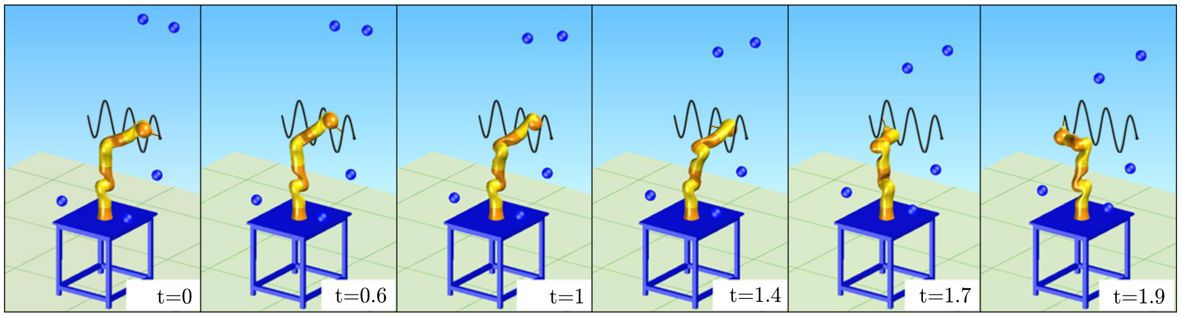


Fig. 4. Purely kinematic planning experiment on the first scenario: sample frames from the solution.

V. Planning Experiments

Our planner was implemented on a 64-bit Intel Core i5-2320 CPU running at 3 GHz using Kite, a software development kit for motion planning currently marketed by Siemens. In this section, we report results for three scenarios involving a 3-dimensional tip positioning task for a KUKA LWR-IV 7-DOF manipulator; the degree of redundancy for this kind of task is 3 (the wrist roll is frozen as it does not contribute to tip positioning). Joint velocity and torque limits for this robot were taken from the official documentation. The paper video attachment shows clips of the generated motions.

We used the same settings in all scenarios. In particular, a sequence of $N = 11$ equispaced samples are taken from the desired task path (including the endpoints, which correspond to $s = 0$ and $s = 1$). In the motion generation scheme, we use $\boldsymbol{K}_p = \boldsymbol{K}_d = 400 \cdot \boldsymbol{I}$, and the null space term is constrained to be in norm at most 10% of the range space term. Integration is performed using Euler method with step size $\Delta s = 0.002$. The only variable parameter was the upper bound for $|\ddot{s}|$; in particular, we used $c_{\max} = 0.1$ in the first scenario, $c_{\max} = 1$ in the second and $c_{\max} = 2$ in the third.

In the first scenario, the manipulator is mounted on a table and must move its tip along a planar sinusoidal path, while avoiding collisions with the table and with spherical obstacles that move back and forth along line segments. Figure 3 shows a dynamically feasible solution computed by the proposed planner (DF_TCMP_MO). For comparison, Figure 4 shows the solution computed by the purely kinematic planner TCMP_MO proposed in [15], which does not consider torque bounds. Not surprisingly, the dynamically feasible plan is slightly slower, as shown by Figure 5, which shows the time history along the solutions. However, the required torques with DF_TCMP_MO are always within the bounds, whereas the solution computed by TCMP_MO exceeds the torque limits (see Figure 6).

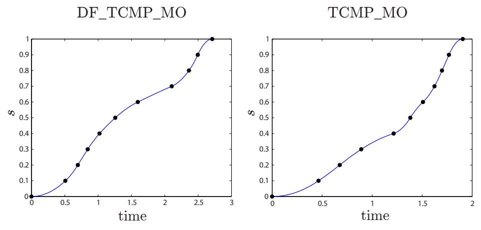


Fig. 5. Planning experiments on the first scenario: Time histories.

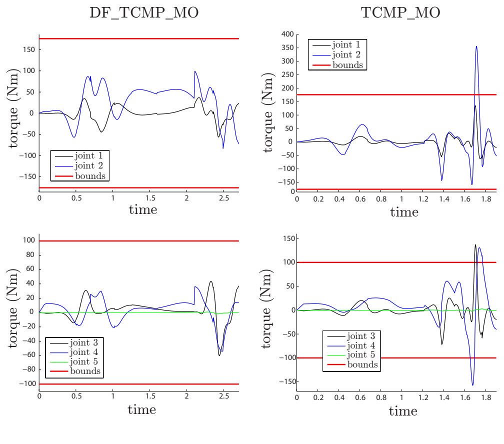


Fig. 6. Planning experiment on the first scenario: Required torques.

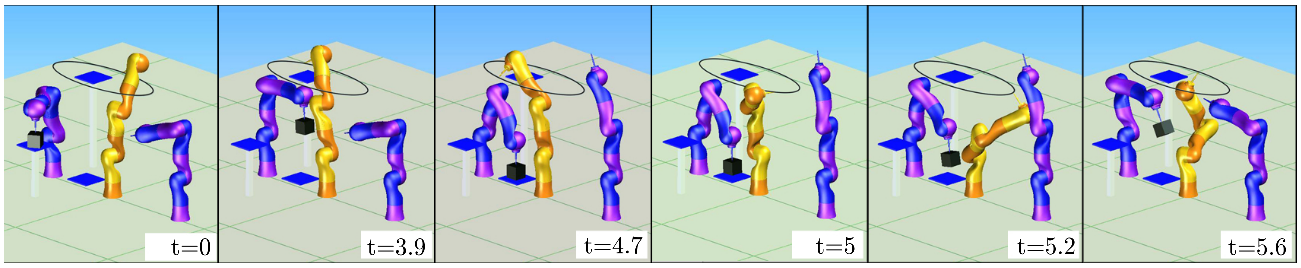


Fig. 7. Dynamically feasible planning experiment on the second scenario: sample frames from the solution.

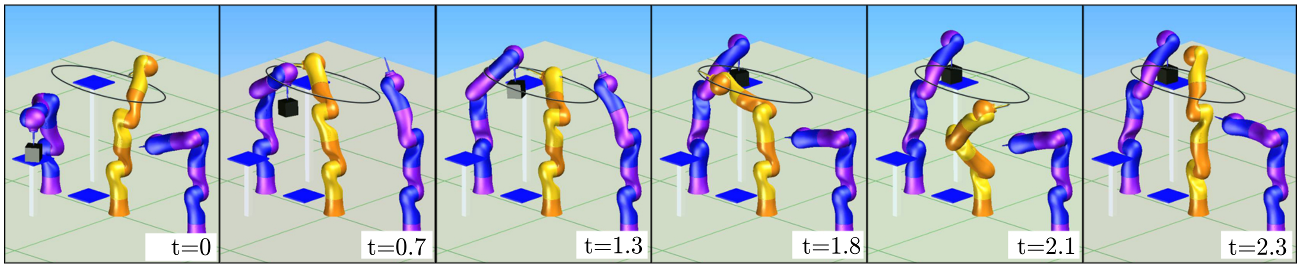


Fig. 8. Purely kinematic planning experiment on the second scenario: sample frames from the solution.

In the second scenario, the manipulator must move its tip along a planar elliptic path while avoiding collisions with other two identical manipulators that execute repetitive tasks. Figures 7 and 8 respectively show a dynamically feasible and a purely kinematic solution found for this scenario. The corresponding time histories and torque profiles are reported in Figures 9 and 10. Once again, the dynamically feasible solution is slower but the torque requirements of the purely kinematic solution, which includes a very fast backward/forward movement, cannot be met.

In the third scenario, the manipulator must move its tip along a circular path while avoiding a ball that goes back and forth along the same path. Due to the particular motion of the ball, any solution must include a backward/forward movement to avoid a collision (see the video attachment). In fact, the solution computed by the DF_TCMP_MO planner and shown in Figure 11 contains two motion reversals, as confirmed by the associated time history (Figure 12). The torque profiles are again feasible, see Figure 13.

Table I collects some details (execution time, number of nodes, etc) about the dynamically feasible solutions found for the three scenarios by the DF_TCMP_MO planner. See also `http://www.diag.uniroma1.it/labrob/research/TCMP_MO.html`.

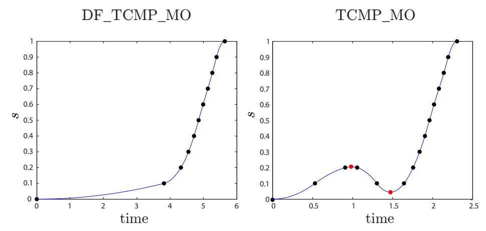


Fig. 9. Planning experiments on the second scenario: Time histories.

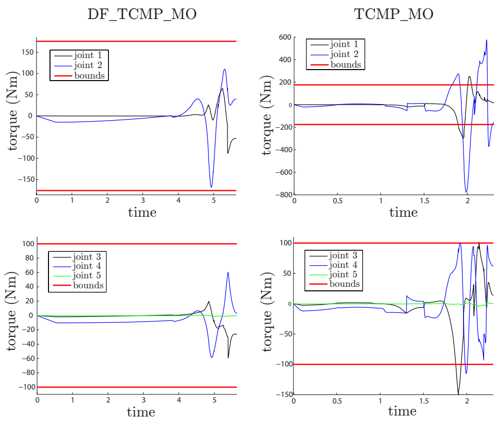


Fig. 10. Planning experiment on the second scenario: Required torques.

VI. Conclusions

We have presented a randomized algorithm for planning dynamically feasible motions of robots subject to geometric task constraints in the presence of moving obstacles. The proposed method builds upon our previous results on task-constrained motion planning with moving obstacles. With respect to our previous formulation, the inclusion of bounds on the available actuator torques leads to the adoption of an acceleration-level motion generation scheme. As a consequence, the new planner operates in a task-constrained state space extended with time. The generated trajectories are collision-free, obey velocity and torque bounds, and satisfy the task constraint with arbitrary accuracy. The effectiveness of the proposed planner has been shown in three scenarios of different complexity, all involving a 7-dof manipulator.

We are currently working to extend the proposed approach to the case of underactuated robots. The challenge of un-

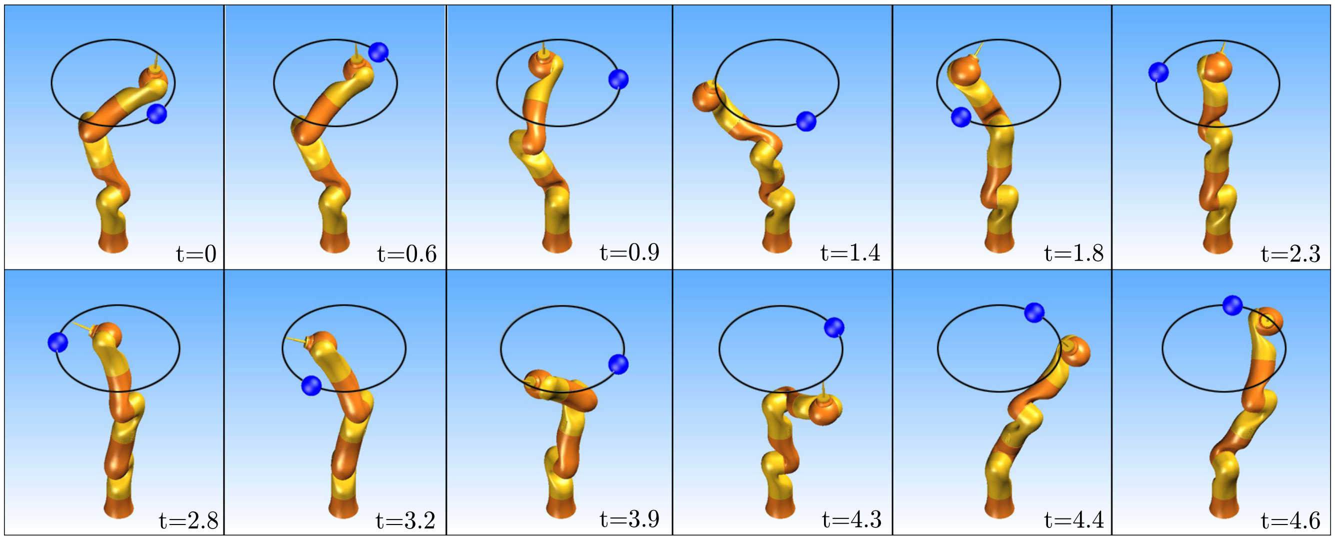


Fig. 11. Dynamically feasible planning experiment on the third scenario: sample frames from the solution.

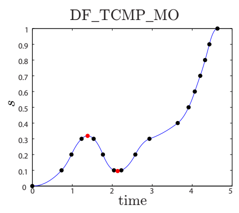


Fig. 12. Planning experiment on the third scenario: Time history.

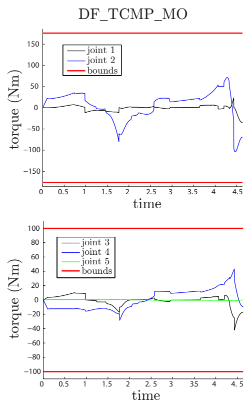


Fig. 13. Planning experiment on the third scenario: Required torques.

deractuation, which is naturally present in many advanced robotic mechanisms (e.g., robotic hands, humanoids, UAVs), is obviously that arbitrary generalized accelerations cannot be produced. Other current work is aimed at relaxing the assumption of fully known obstacle motion, by developing an on-line version of the proposed planner that uses predictions computed on the basis of sensory information.

TABLE I

exp	exec time	vertexes	coll checks	duration	mean task error
scenario 1	510 s	766	274345	2.71 s	5.8 mm
scenario 2	417 s	425	100312	5.63 s	6.4 mm
scenario 3	437 s	985	299497	4.63 s	5.4 mm

REFERENCES

[1] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," *J. of the ACM*, vol. 41, no. 4, pp. 764–790, 1994.

[2] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[3] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 4, pp. 477–521, 1987.

[4] T. Fraichard, "Dynamic trajectory planning with dynamic constraints: A 'state-time space' approach," in *1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1993, pp. 1393–1400.

[5] P. Fiorini and Z. Shiller, "Time optimal trajectory planning in dynamic environments," in *1996 IEEE Int. Conf. on Robotics and Automation*, Minneapolis, MN, 1996, pp. 1553–1558.

[6] ——, "Motion planning in dynamic environments using velocity obstacles," *Int. J. of Robotics Research*, vol. 17, pp. 760–772, 1998.

[7] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," in *2001 IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, 2001, pp. 3716–3721.

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

[9] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[10] L. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning high-dimensional configuration spacess," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

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

[12] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Trans. on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[13] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, 2009, pp. 625–632.

[14] 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.

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