2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Task-Constrained Motion Planning with Moving Obstacles

Massimo Cefalo, Giuseppe Oriolo, Marilena Vendittelli

Abstract— **We consider the problem of planning the motion of redundant robotic systems subject to geometric task constraints in the presence of obstacles moving along known trajectories. Building on our previous results on task-constrained motion planning, we propose a control-based motion planner that works directly in the task-constrained configuration space extended with the time dimension. The generated trajectories are collision-free and satisfy the task constraint with arbitrary accuracy. Bounds on the achievable generalized velocities may also be taken into account. The proposed approach is validated through planning experiments on a 7-dof articulated robot and an 8-dof mobile manipulator.**

I. Introduction

The basic formulation of the motion planning problem assumes that the robot workspace is populated by static obstacles. A particularly relevant extension in real world applications allows for the presence of obstacles that move along trajectories whose predictability can range from fully known to completely unknown [1].

In this work, we focus on the motion planning problem with obstacles moving along known trajectories. This has been shown to be a computationally difficult problem even for a single rigid body with unbounded velocity in [2]. Early solutions (like, e.g., [3], [4], [5]) extend combinatorial or sampling-based methods by considering a planning space consisting of the configuration or state space augmented with the time dimension (respectively, configuration-time and state-time space). Other ad-hoc methods include the velocity obstacle technique proposed in [6], [7], [8].

All the above methods prove to be prohibitively inefficient when dealing with articulated robotic systems, due to the computational complexity of the problem. A viable alternative is the randomized sampling-based technique proposed in [9], where kinematic and dynamic constraints are considered and an estimate of the rate of convergence is provided.

Between the antithetical assumptions of complete knowledge and complete ignorance of the obstacle motion, one may take an intermediate viewpoint which considers an uncertain motion model. For example, two kinds of uncertainties are taken into account in [10], namely on the obstacle path and on the time history along such path. Using this stochastic description, a probability distribution for the obstacles motion is computed and used to plan the robot trajectory accordingly.

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,vendittelli}@dis.uniroma1.it. Work supported by the EU FP7 ICT-287513 SAPHARI project.

The above cited works solve the motion planning problem from a start to a goal configuration, and do not consider motion constraints induced by tasks, which in general restrict the set of feasible configurations to a lower-dimensional submanifold of the configuration space. Nevertheless, task constraints constantly arise in robotic operation; notable examples include manipulation, locomotion, visual tracking, drawing, cutting, welding, opening doors, and so on. In the current paradigm of ubiquitous robotics, these tasks must be executed in unstructured, dynamic environments, possibly shared with human beings. It is therefore of central importance to lay the basis for methods that can plan the motion of robotic systems subject to task constraints in the presence of moving obstacles.

A complication of Task-Constrained Motion Planning (TCMP) is that random sampling of the configuration space is no more effective, because the probability of generating a sample in the feasible submanifold is zero. To address this issue, a popular approach in the literature is to generate samples using standard randomized search algorithms such as, e.g., PRM [11] or RRT [12], and then projecting the samples on the submanifold with a given error tolerance. This projection may be performed via randomized gradient descent, tangent space sampling or retraction [13].

A conceptually different approach that is not based on projection was proposed in [14], where we introduced a control-based method for TCMP guaranteeing continuous satisfaction of the task constraint and probabilistic completeness. With respect to projection-based methods, and also to our previous TCMP planners [15], [16], the control-based planner allows to arbitrarily improve the task accuracy without increasing the roadmap complexity.

In all these works on TCMP, however, the obstacles are static. In this paper, we shall consider Task Constrained Motion Planning with Moving Obstacles, or TCMP_MO. In particular, we will assume that the obstacle trajectories are known in advance, as a first step towards the solution of more realistic problems with reduced predictability levels of obstacle motion. Moreover, there exist scenarios or fields of applications in which the trajectories of obstacles are actually known in advance. This is true, for example, in some industrial robotics applications, or in the animation of digital characters.

Here, we extend the method of [14] to the TCMP_MO problem by augmenting the task-constrained planning space with the time dimension and then growing an RRT in the collision-free part of this space. To the best of our knowledge, ours is the first task-constrained motion planner

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

that can handle moving obstacles. Moreover, kinematic and dynamic constraints such as velocity and torque limits can be incorporated in the solution.

The paper is organized as follows. Section II formally defines the TCMP_MO problem. Section III discusses the structure of the planning space, while Sections IV and V describe the TCMP_MO motion planner. Section VI describes how the proposed method can be extended to take into account limits on the achievable generalized velocities. Planning experiments are two commercial robots are presented in Section VII. Possible future work is mentioned in the concluding section.

II. Formulation of the TCMP_MO Problem

Consider a robot with configuration $\boldsymbol{q}$ and denote by $\mathcal{C}$ its n_q-dimensional configuration space. The robot lives in a workspace $\mathcal{W} \subset I\!R^3$ which is populated by moving obstacles. Denote by $\mathcal{R}(\boldsymbol{q}) \subset \mathcal{W}$ and $\mathcal{O}(t) \subset \mathcal{W}$, respectively, the volume occupied by the robot at configuration $\boldsymbol{q}$ and by the obstacles at time t.

Task coordinates $\boldsymbol{y}$ take values in an n_y-dimensional task space $\mathcal{Y}$ and are related to the configuration coordinates by a nonlinear map

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}). \tag{1}$$

We take the following assumptions.

- A1 The robot is kinematically redundant for the task $\boldsymbol{y}$, i.e., $n_q > n_y$.
- A2 We have full a priori information on the obstacle motion, i.e., $\mathcal{O}(t) \subset \mathcal{W}$ is known for all t.

Now suppose that a desired path $\boldsymbol{y}_d(s)$, $s \in [s_i, s_f]$, is assigned for the task variables $\boldsymbol{y}$. The Task-Constrained Motion Planning with Moving Obstacles (TCMP_MO) problem consists in searching for a configuration space trajectory of the robot such that the geometric path underlying the trajectory is consistent with the assigned path and all collisions are avoided.

Clearly, a solution to the TCMP_MO problem consists of two components: a configuration space task path, and a time history along this path. This leads to the following rigorous formulation (compare with the TCMP problem in [14]).

> Under assumptions A1–A2, consider a desired task path $\boldsymbol{y}_d(s) \in \mathcal{Y}$, $s \in [s_i, s_f]$. A solution to the TCMP_MO problem is 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 $s(T) = s_f$;
> 2. for all $t \in [0, T]$, it is $\boldsymbol{y}(t) = \boldsymbol{f}(\boldsymbol{q}(s(t))) = \boldsymbol{y}_d(s(t))$;
> 3. for all $t \in [0, T]$, it is $\mathcal{R}(\boldsymbol{q}(s(t))) \cap \mathcal{O}(t) = \emptyset$.

Condition 1 requires the robot to start/stop at the beginning/end of the task path, condition 2 guarantees that the task constraint is always satisfied, and condition 3 entails avoidance of the moving obstacles (self-collisions can be easily incorporated and in fact they are included in our implementation).

Continuity of the time history $s(t)$ is necessary to guarantee that $\boldsymbol{y}$ does not 'jump' along the assigned task path. In addition, continuity together with condition 1 ensures that all values of s in $[s_i, s_f]$ are generated.

An important feature of TCMP_MO is that $s(t)$ is not required to be non-decreasing: s must start at s_i and end at s_f, but this increase is not required to be monotonic during the motion. That is, at any point along the path, s may increase (*forward motion*), remain constant (*self-motion*) or even decrease (*backward motion*), if these maneuvers are useful for avoiding moving obstacles. In other words the assigned $\boldsymbol{y}_d(s)$, $s \in [s_i, s_f]$, will only be the 'footprint' of the motion[1], whereas the actual motion $\boldsymbol{y}_d(t)$, $t \in [0, T]$, will depend on the choice of $s(t)$.

Finally, note that both the final configuration $\boldsymbol{q}(s_f)$ and the final time T are not specified in advance and will be a byproduct of the solution. This is a distinctive aspect of our approach with respect to [9].

III. The Planning Space

Due to the presence of moving obstacles, the planning space for TCMP_MO is not a simple subset of the configuration space $\mathcal{C}$. A configuration may be, in fact, admissible at a certain time instant and not admissible at another due to obstacle movement. Hence, we need to include time in the picture.

In particular, define:

- the *configuration-time space* (henceforth CTS) as
$$\mathcal{S} = \mathcal{C} \times [0, \infty);$$
- the *occupied CTS* as
$$\mathcal{S}_{\text{occ}} = \{(\boldsymbol{q}, t) \in \mathcal{S} : \mathcal{R}(\boldsymbol{q}(t)) \cap \mathcal{O}(t) \neq \emptyset\};$$
- the *free CTS* as
$$\mathcal{S}_{\text{free}} = \mathcal{S} \setminus \mathcal{S}_{\text{occ}};$$
- the *task-constrained CTS* as
$$\mathcal{S}_{\text{task}} = \{(\boldsymbol{q}, t) \in \mathcal{S} : \boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{y}_d(s), s \in [s_i, s_f]\}.$$

The set $\mathcal{S}_{\text{task}}$ is a manifold with boundary that naturally decomposes as a foliation:

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

with the generic *leaf* defined as

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

On each leaf, t can assume any value in $[0, \infty)$. Figure 1 illustrates the structure of $\mathcal{S}_{\text{task}}$: the set of configurations satisfying the task constraint at s (the so-called *self-motion*

[1] It should be noted that the above TCMP_MO formulation (assigned task *path* and known obstacle trajectories) is the only interesting instance of TCMP with moving obstacles. To plan a motion among moving obstacles while constrained to a task *trajectory* one can directly use the method of [14]. In fact, assigning a task trajectory implies that the time history along the configuration space path is given and therefore the robot cannot retract, stop, slow down or accelerate on the path to avoid obstacles.

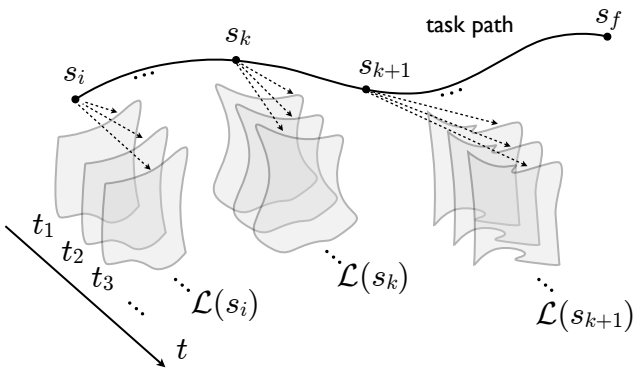

Fig. 1. The structure of $\mathcal{S}_{\text{task}}$: each leaf $\mathcal{L}(s)$ is the set of configurations satisfying the task constraint at s replicated along the time axis.

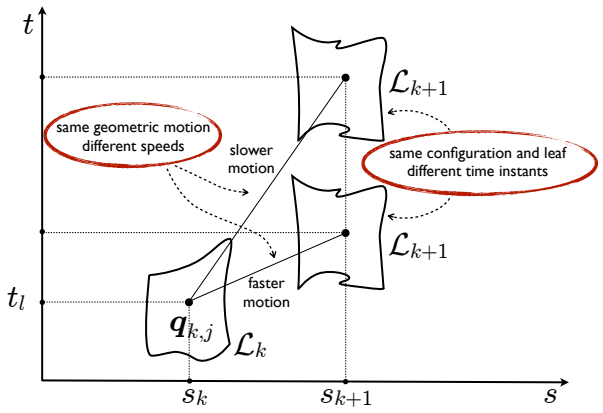

Fig. 2. Generation of forward motions in TCMP_MO: different choices of $\dot{s} > 0$ produce the same geometric motion at different speeds. The same is true for backward motions associated to $\dot{s} < 0$ (not shown).

manifold associated to the task value at s) is replicated along the time axis to form a leaf of $\mathcal{S}_{\text{task}}$.

The existence of a solution to the TCMP_MO problem depends on the motion of the obstacles, and in particular on the connectedness of $\mathcal{S}_{\text{task}} \cap \mathcal{S}_{\text{free}}$, i.e., the portion of the free CTS that is compatible with the task path constraint.

IV. Motion Generation

The basic element of our planner is a motion generation scheme that produces subpaths contained in $\mathcal{S}_{\text{task}}$. The idea is to extend the motion generation scheme of [14] so as to plan, in addition to a geometric motion $\boldsymbol{q}(s)$, also a time history $s(t)$. This is obviously required because collisions with moving obstacles can only be checked in the time domain. In particular, using the notation $(\,)' = d(\,)/ds$, we have

$$\dot{\boldsymbol{q}} = \boldsymbol{q}'\dot{s} \tag{2}$$

and therefore we will generate $\boldsymbol{q}'$ (the tangent vector in configuration space) and $\dot{s}$ (the variation rate of parameter s) separately.

One possibility is to choose $\dot{s}$ first. In particular, our generation of $\dot{s}$ is primitive-based:

$$\dot{s} \in \{-c_{\max}, \ldots, -c_1, 0, c_1, \ldots, c_{\max}\} \tag{3}$$

where $0 < c_1 < \ldots < c_{\max}$. The resulting profile $s(t)$ will be continuous and piecewise-linear: choosing $\dot{s} > 0$ will produce a forward motion along the task path, $\dot{s} = 0$ will result in a self-motion (i.e., a motion that does not change the value of the task variable), whereas $\dot{s} < 0$ will produce backward motions along the task path. In our implementation, we always generate a triplet, i.e., a positive, a negative and a zero $\dot{s}$.

Once $\dot{s}$ is chosen, it is necessary to generate $\boldsymbol{q}'$. For example, consider the case of a forward motion. We let

$$\boldsymbol{q}' = \boldsymbol{J}^{\dagger}(\boldsymbol{y}'_d + \boldsymbol{K}\,\boldsymbol{e}_y) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})\boldsymbol{w}, \tag{4}$$

where $\boldsymbol{J}^{\dagger}$ is the pseudoinverse[2] of the task Jacobian matrix $\boldsymbol{J} = \partial \boldsymbol{f}/\partial \boldsymbol{q}$, $\boldsymbol{K}$ is a positive definite gain matrix, vector $\boldsymbol{e}_y = \boldsymbol{y}_d - \boldsymbol{y}$ is the task error, $\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ is the orthogonal projection matrix in the null space of $\boldsymbol{J}$, and $\boldsymbol{w}$ is an arbitrary n_q-vector acting as a *residual* input that produces internal motions with no effect on the task. Forward motion is then generated by integrating eq. (4) over s.

A similar formula is used for a backward motion. In particular, this is generated by backward integration of eq. (4) where $\boldsymbol{y}'_d$ is replaced with $-\boldsymbol{y}'_d$ and the value of $\boldsymbol{y}_d(s)$ used to compute the task error $\boldsymbol{e}_y$ is sampled from the assigned task path by going 'backwards' from the current value of s.

Finally, for a self-motion, we simply set $\boldsymbol{y}'_d = \boldsymbol{0}$ and integrate (4) forward in an auxiliary parameter σ, with $\sigma \in [0, \sigma_{\max}]$. This will result in a self-motion of duration $\sigma_{\max}$.

V. TCMP_MO Planner

The proposed planner builds an RRT in the admissible search space $\mathcal{S}_{\text{task}} \cap \mathcal{S}_{\text{free}}$. For the construction of the tree we make use of samples of the desired task path $\boldsymbol{y}_d(s)$. In particular, denote by $\{s_1 = s_i, s_2, \ldots, s_{N-1}, s_N = s_f\}$ a predefined sequence of N path parameter values, by $\boldsymbol{y}_k = \boldsymbol{y}_d(s_k)$ the sample of the path corresponding to s_k (note that we drop the d subscript) and by $\mathcal{L}_k = \mathcal{L}(s_k)$ the associated leaf (see Fig. 1).

Every vertex of the tree is a sample point of $\mathcal{S}_{\text{task}}$. More precisely, each vertex belongs to a certain leaf $\mathcal{L}_k$, $k = 1, \ldots, N$, and consists of a configuration $\boldsymbol{q}$ such that $\boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{y}_k$ and of a time t at which $\boldsymbol{q}$ has been reached. The tree edges are collision-free subtrajectories obtained by applying the motion generation scheme (3–4) starting from vertexes. Construction of the tree proceeds as follows.

At each iteration, a configuration $\boldsymbol{q}_{\text{rand}}$ is first generated by picking a random value from the sequence $\{s_1, s_2, \ldots, s_{N-1}, s_N\}$ and an inverse kinematic solution for the corresponding task sample. A time instant t_{rand} is then attached to $\boldsymbol{q}_{\text{rand}}$ by random choice in $[0, t_{\max}]$, where $t_{\max}$ denotes the largest time associated to a vertex of the tree so far. By construction, $(\boldsymbol{q}_{\text{rand}}, t_{\text{rand}})$ is a sample of $\mathcal{S}_{\text{task}}$.

The tree is then searched to find the vertex $(\boldsymbol{q}_{\text{near}}, t_{\text{near}})$

[2]The proposed planner discards configurations where $\boldsymbol{J}$ loses rank, so that we always have $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}$.

that is closest[3] to $(\boldsymbol{q}_{\rm rand}, t_{\rm rand})$. Call s_k and $\mathcal{L}_k$ the parameter value and leaf, respectively, corresponding to $\boldsymbol{q}_{\rm near}$; also, rename $\boldsymbol{q}_{\rm near}$ as $\boldsymbol{q}_{k,j}$ and $t_{\rm near}$ as t_l. Note that index j is used to distinguish the different configurations at which the tree may have reached the same $\mathcal{L}_k$.

At this point, the tree is extended from $(\boldsymbol{q}_{k,j}, t_l)$. First, a triplet of values of $\dot{s}$ is chosen from (3) and the corresponding time histories $s(t)$ are reconstructed within the current subinterval, stopping (1) for $\dot{s} > 0$, when we have reached s_{k+1}; (2) for $\dot{s} < 0$, when we have reached s_{k-1}; (3) for $\dot{s} = 0$, after $\sigma_{\max}$ seconds. As a byproduct, we obtain the time instants associated to the new vertexes. Then, a fixed number of different constant values are randomly generated for the residual vector $\boldsymbol{w}$, with the constraint that the norms of the two terms in the rhs of eq. (4) are in a certain proportion. For each value of $\boldsymbol{w}$, three subpaths are generated by applying the motion generation scheme from $\mathcal{L}_k$ to $\mathcal{L}_{k+1}$ (forward motion), from $\mathcal{L}_k$ to $\mathcal{L}_{k-1}$ (backward motion) or on $\mathcal{L}_k$ (self-motion). During the integration, the corresponding subtrajectories $\boldsymbol{q}(s(t))$ are checked for collision with the moving obstacles. For each category of motion (forward, backward and self-), only the collision-free subtrajectory terminating closest to $\boldsymbol{q}_{\rm rand}$ is retained.

Figure 2 illustrates how the TCMP_MO planner works in the case of a forward motion. Once a vertex $(\boldsymbol{q}_{k,j}, t_l)$ on $\mathcal{L}_k$ has been selected for extension, different choices of $\dot{s} > 0$ in (3) for the same tangent vector $\boldsymbol{q}'$ would produce paths terminating in the same configuration on $\mathcal{L}_{k+1}$ at different time instants.

VI. Incorporating Velocity Limits

A practically important extension of the TCMP_MO problem is obtained by including in its formulation constraints of the form $|\dot{q}_i| \leq \dot{q}_{i,M}$, $i = 1, \ldots, n_q$. These generalized velocity bounds, invariably present in actual robotic systems, are obviously relevant because they limit the robot ability of speeding up along the path.

The straightforward approach to take velocity bounds into account would be to check during the integration whether $\dot{\boldsymbol{q}} = \boldsymbol{q}'\dot{s}$ is admissible, discarding the current subtrajectory otherwise. This validation procedure may however be inefficient and lead to a large number of rejected motions.

A more efficient approach is to directly generate trajectories which are guaranteed to be feasible. To this end, one may simply revert the motion generation sequence: generate the geometric path first, delaying any collision check, and then choose a value for $\dot{s}$ so as to comply with the velocity constraints. For simplicity, consider a version of the planner in which self-motions are not generated. The tree expansion procedure from $(\boldsymbol{q}_{k,j}, t_l)$ becomes the following (compare with the previous section):

1) randomly choose a certain number of values for the residual input $\boldsymbol{w}$;
2) for each value of $\boldsymbol{w}$, generate a forward motion and a backward motion;
3) for each category, retain only the subpath terminating closest to $\boldsymbol{q}_{\rm rand}$;
4) for each subpath:
   a) compute
   $$c_{\max} = \min_{i=1,\ldots,n_q} \frac{\dot{q}_{i,M}}{q'_{i,\max}},$$
   where $q'_{i,\max}$=$\max_{s\in\mathcal{I}} |q'_i(s)|$ and $\mathcal{I} = [s_k, s_{k+1}]$ for a forward motion, $\mathcal{I} = [s_k, s_{k-1}]$ for a backward motion.
   b) choose $\dot{s}$ (positive for a forward motion, negative for a backward motion) as in eq. (3) using the above $c_{\max}$;
   c) check the resulting subtrajectory for collisions.

This expansion mechanism produces subtrajectories that are always compliant with the velocity constraints. The side effect is that collision check, which can only be performed once a time history has been chosen, must now be delayed to the end of the procedure.

The above procedure may be easily extended to allow the generation of admissible self-motions by appropriately choosing the $\sigma_{\max}$ parameter.

VII. Planning Experiments

We implemented the proposed TCMP_MO planner in Kite (a cross-platform software for motion planning produced by Kineo CAM) on a 64-bit Intel Core i5-2320 CPU running at 3 GHz. We report planning results for two KUKA robots: an LWR-IV 7-DOF articulated robot and a youBot mobile manipulator. Animations of the generated motions are contained in the video clip accompanying this paper.

Figure 3 shows the first planning scenario. An LWR-IV manipulator mounted on a table must move its tip along a sinusoidal path contained in a vertical plane, while avoiding collisions with the table, with itself and with five moving obstacles. These are balls moving back and forth along line segments; the time history of each ball along its path is a sinusoid of different frequency. The degree of redundancy is 3 (the task is 3-dimensional and the wrist roll is frozen because it does not contribute to positioning the tip). Joint velocity limits taken from the actual robot specifications are enforced by the previous technique.

In the second scenario, shown in Fig. 4, a youBot must move its tip along an ellipse lying on a plane slightly tilted w.r.t. the horizontal plane, while avoiding collisions with itself and with five balls that roll back and forth on the floor. Since the mobile base of the youBot is omnidirectional, and the wrist roll of the 5-DOF arm is again frozen, the degree of redundancy for this case is $3 + 4 - 3 = 4$. Velocity limits are enforced also in this case.

We used the same planner settings in both scenarios. In particular, we extract $N = 11$ equispaced samples[4] from each desired task path (including the endpoints, which correspond to $s = 0$ and $s = 1$), while motion is generated letting $c_{\max} = 0.15$ in (3) and $\boldsymbol{K} = 100 \cdot \boldsymbol{I}$ in (4), where the

[3] A weighted metric that accounts for distances in the configuration-time space is used here.

[4] See [14] for a discussion on the choice of N.

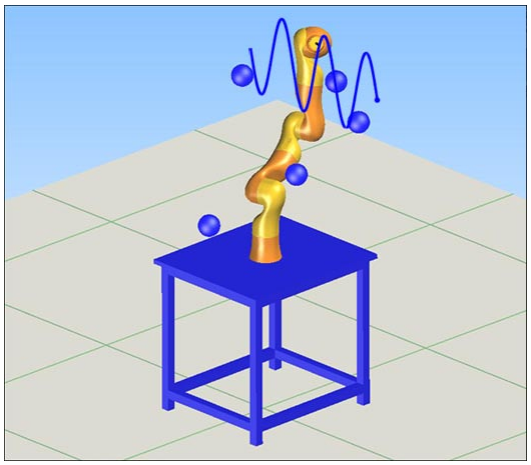

Fig. 3. First planning scenario.

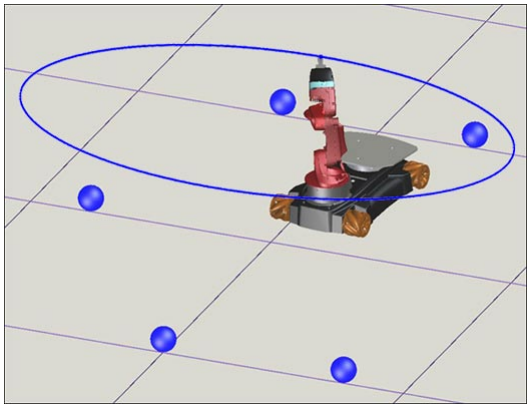

Fig. 4. Second planning scenario.

null space term is constrained to be in norm at most 200% of the range space term. Integration is performed numerically using Euler's method with step size $\Delta s = 0.002$.

The solution for the first scenario summarized in Fig. 5 took about 70 s to compute. Other details about the solution are given in Table I. Note in particular the duration of the motion, which is a byproduct of the solution, and the negligible error along the whole task path.

A solution for the second scenario[5] is depicted in Fig. 6. As shown by Table I, a significantly longer computation was needed due to the higher dimension of the configuration space. The mean task error is again minimal.

Figure 7 shows the time histories along the assigned path for the two solutions. Due to the primitive-based choice (3) of $\dot{s}$, a piecewise-linear continuous profile is always obtained. Each black dot in the plot relates a value of s to the time instant t at which the corresponding leaf has been reached; segments between dots are associated to elementary subpaths. Incidentally, both solutions contain a back-and-forth motion: the first from the leaf corresponding to $s = 0.8$, and the second from the leaf corresponding to $s = 0.5$. Also, self-motions do not appear in either solution.

[5]One may notice that, although the assigned task path is closed, the generated motion in configuration space is not cyclic. If the task is to be repeated, cyclicity may be enforced by extending the planning approach of [17] to the case of moving obstacles.

TABLE I

exp	exec time	vertexes	coll checks	duration	mean task error
LWR-IV	70 s	82	43249	16.24 s	0.41 mm
youBot	360 s	2022	905230	20 s	0.15 mm

VIII. Conclusion

For redundant robotic systems subject to geometric task constraints, we have presented a motion planning method that can handle obstacles moving along a priori known trajectories. Our planner guarantees continuous satisfaction of the task constraints and may also accommodate bounds on the achievable generalized velocities. Planning experiments on two commercial robots (an articulated arm and a mobile manipulator) have been presented to show the performance of the proposed method.

One direction for future work will be considering the dynamics capabilities of the robot, e.g., torque limits. In the presence of moving obstacles, time scaling on a planned trajectory so as to stay within the torque limits is not a viable option, because the scaled motion will not be collision-free in general. The only possible solution is to incorporate the robot dynamics into the planner.

Other extensions of the present approach will be aimed at relaxing the assumption of known obstacle trajectories. In particular, our ultimate goal is to devise an on-line version of the present planner based on sensor predictions of the obstacle motion.

References

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

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

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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, May 2001, pp. 3716–3721.

[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] J. Miura and Y. Shirai, "Modeling motion uncertainty of moving obstacles for robot motion planning," in *2000 IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, April 2000, pp. 2258–2263.

[11] 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.

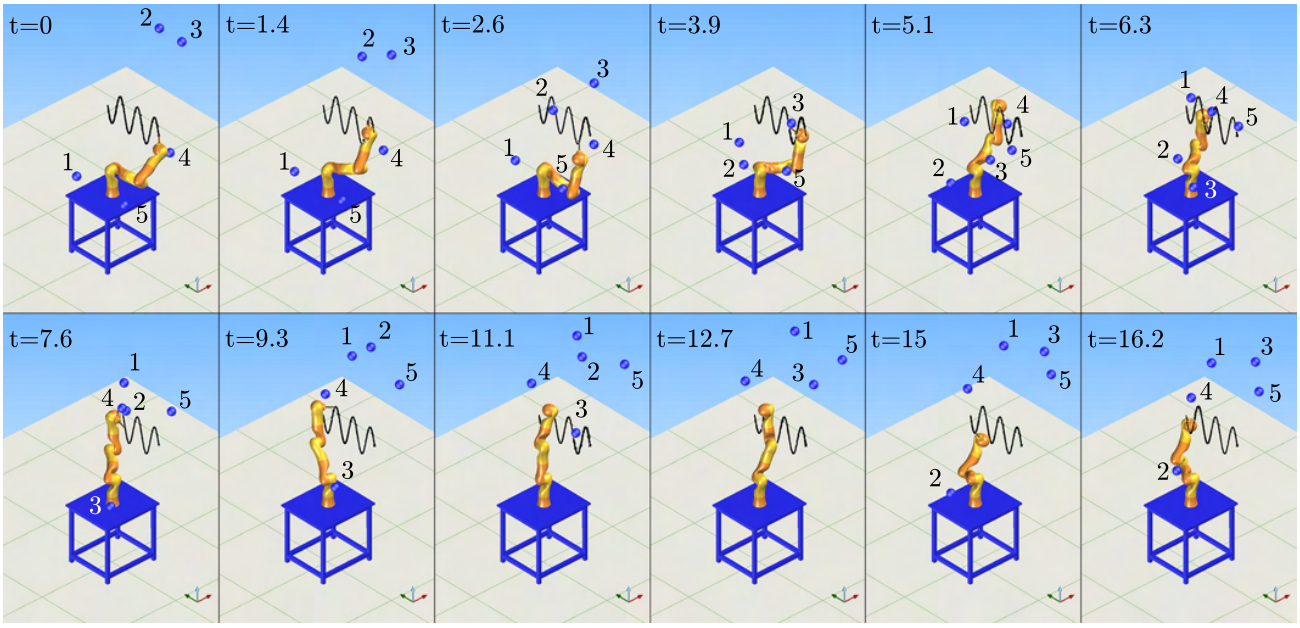

Fig. 5. Planning experiment on the LWR-IV: sample frames from the solution.

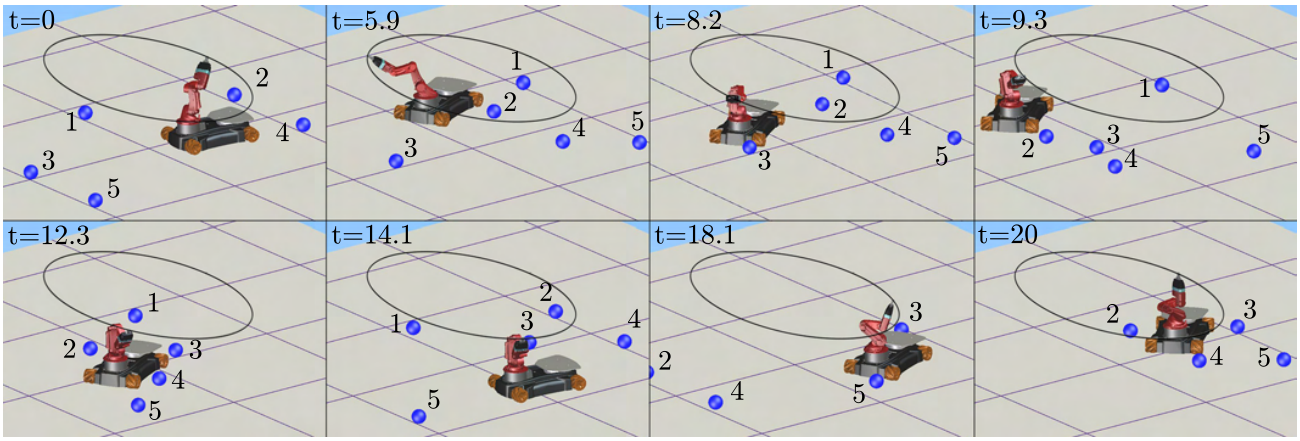

Fig. 6. Planning experiment on the youBot: sample frames from the solution.

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

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

[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] 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.

[16] 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.

[17] M. Cefalo, G. Oriolo, and M. Vendittelli, "Planning safe cyclic motions under repetitive task constraints," in *2013 IEEE Int. Conf. on Robotics and Automation*, Karlsruhe, DE, May 6 - 10 2013.

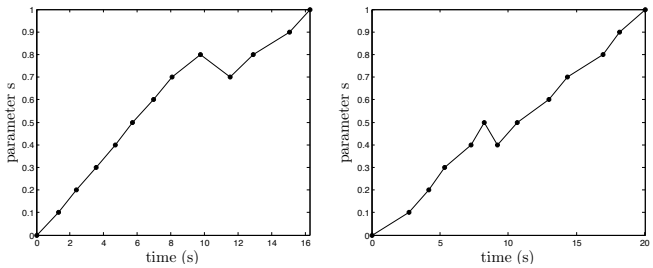

Fig. 7. Planning experiment on the LWR-IV: Time histories along the planned paths. LWR-IV experiment (left) and youBot experiment (right).