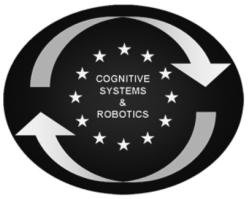
SAPHARI

Safe and Autonomous Physical Human-Aware Robot Interaction



Project funded by the European Community's 7th Framework Programme (FP7-ICT-2011-7)
Grant Agreement ICT–287513

Deliverable D7.4.1

Planning of smooth motion-force transition tasks

Deliverable due date: 31 December 2013	**Actual submission date:** 15 January 2014
Start date of project: 1 November 2011	**Duration:** 48 months
Lead beneficiary: UNIROMA1	**Revision:** Final

Nature: R	**Dissemination level:** PU
R = Report P = Prototype D = Demonstrator O = Other	PU = Public PP = Restricted to other programme participants (including the Commission Services) RE = Restricted to a group specified by the consortium (including the Commission Services) CO = Confidential, only for members of the consortium (including the Commission Services)

Executive Summary

This deliverable of work package WP7 deals with a relatively novel problem, namely planning the motion of an articulated robot that needs to execute a task involving the *hybrid* specification of feasible motion (i.e., compatible with robot kinematic and dynamic constraints, and avoiding collisions with environment obstacles) as well as of desired interaction forces (to be exchanged with environment surfaces at selected locations where the robot is in contact).

Hybrid force/motion is indeed a well-established topic in the robot control literature, but the existing control techniques are only able to deal with local decisions and constraints, and miss the generality of the above formulation (e.g., they cannot guarantee a collision-free motion consistent with the global task, even if one exists). On the other hand, motion planning techniques generate complete solutions in known environments, but do not address tasks with dynamic exchanges of forces/torques, especially when these need to be executed in parallel or in sequence with long range motion tasks.

The algorithmic solution developed within SAPHARI consists in a novel extension of a framework called Task-Constrained Motion Planning (TCMP) that combines control-oriented techniques (path tracking, use of the Jacobian null-space in the presence of redundancy, consideration of robot dynamics) to guarantee accuracy and efficiency, with state-of-the-art methods in randomized motion planning (like RRT, suitably modified) to handle the complexity of large but cluttered search spaces.

The class of considered problems includes also specific SAPHARI requirements such as smooth transition from free-space motion to contact situations and vice versa, inclusion of human-aware constraints and objectives in the planning cost, moving obstacles, consideration of manipulators mounted on mobile bases, inclusion of robot velocity, acceleration, or torque limits, possibly varying along the task execution, and so on. All of these features can be addressed by the proposed numerical method, which is here illustrated on two representative case studies.

The approach is currently being tested also on one of the scenarios of the KUKA use case, namely for the planning of fetch and carry of parts by a commissioning mobile manipulator helping operators in assembly tasks. The present work interfaces on one side with the higher-level task planning methods developed in T7.2-T7.3 of WP7, and provides on the other side nominal motion-force references for the interaction controllers of T3.4 in WP3.

Table of contents

1 Introduction

In the real world, robotic systems are usually assigned tasks that are more general than pure motion tasks. A typical robotic application, may it be in an industrial or in a service context, will often involve contacts and interactions with humans and/or with the environment. For such applications, the motion planning problem must be described considering the presence of hybrid motion-force tasks.

Hybrid tasks have been most often considered in literature from the control point of view, while very few works have been proposed in the past to handle these tasks in a planning context. In this document, we describe a novel motion planning algorithm that is precisely designed to address hybrid tasks, in addition to guaranteeing satisfaction of constraints intrinsic to the specific robot (position, velocity and torque limits at the joints) as well as avoidance of workspace obstacles, fixed or moving.

One important aspect of hybrid motion-force tasks is that they are typically intermittent in nature; i.e., they are composed by sequences of motions in free space and motions in contact with the environment. At the transition points, where contact is established or abandoned, it is necessary to guarantee some smoothness conditions to avoid undesirable impact forces or sudden transient errors. The proposed planner is able to comply with these conditions in a natural way.

To illustrate the performance of the proposed motion planning algorithm, some results are presented for a scenario involving a KUKA LWR manipulator executing a hybrid task that involves both free motion and motion in contact with a planar surface.

2 Problem formulation

For a generic robotic system, let $\boldsymbol{q}$ be a n_q-dimensional vector of generalized coordinates representing the robot configuration, and $\mathcal{C}$ be the configuration space. We denote by $\mathcal{X} = \mathcal{C} \times T_{\boldsymbol{q}}\mathcal{C}$, the $n_q \times n_q$ robot state space, where $T_{\boldsymbol{q}}\mathcal{C}$ is the tangent space of $\mathcal{C}$ at $\boldsymbol{q}$, and by $\mathcal{W}$ the subset of $\mathbb{R}^2$ or $\mathbb{R}^3$ representing the robot workspace.

In the general formulation of the problem, we assume that the workspace is populated by fixed and 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 all the obstacles at time t. Throughout the rest of the document, it is assumed that $\mathcal{O}(t)$ is known for all t; i.e., that the obstacle motion is fully predictable. The assumption that the trajectories of the moving obstacles are known in advance is a first step towards the solution of more realistic problems with reduced predictability levels of obstacle motion. Note that there exist scenarios or fields of applications in which this assumption is actually verified. This is true, for example, in some industrial robotics applications, or in the animation of digital characters.

2.1 Robot constraints

In a planning problem, we must also take into account a number of limitations intrinsic to the considered robot. Constraints that are invariably present in robotic systems are position constraints (joint limits), kinematic constraints (velocity limits) and dynamic constraints (torque limits). The first two are expressed as upper bounds on the absolute value of joint positions and velocities

$$|\boldsymbol{q}| \leq \boldsymbol{q}_M, \qquad |\dot{\boldsymbol{q}}| \leq \dot{\boldsymbol{q}}_M, \tag{1}$$

whereas dynamic constraints limit the available actuator generalized forces τ (henceforth simply referred to as *torques*):

$$|\boldsymbol{\tau}| \leq \boldsymbol{\tau}_M. \tag{2}$$

We call *feasible* a trajectory in $\boldsymbol{q}(t)$ in $\mathcal{C}$ that satisfies constraints (1–2).

2.2 Motion-force task definition

We assume that the robot is assigned a general task that prescribes the motion of a specific point of the kinematic chain (typically, the end-effector) as well as the forces/moments that the robot exchanges with the environment. In the following, we call this a *motion-force task* for brevity. Note that the existence of a force task implies that the motion task brings the robot in contact with one or more surfaces belonging to the boundary of the obstacle region, at least for part of the motion.

Denote by $\boldsymbol{y}$ the *motion task* coordinates, which take values in an n_y-dimensional space $\mathcal{Y}$, and by $\boldsymbol{f}$ the *force task* coordinates, which take values in an n_f-dimensional task space $\mathcal{F}$ (remember that $\boldsymbol{f}$ may include moments).

The motion task coordinates $\boldsymbol{y}$ are related to the generalized coordinates $\boldsymbol{q}$ by the forward kinematic map

$$\boldsymbol{y} = \boldsymbol{k}(\boldsymbol{q}), \tag{3}$$

whose differential version is

$$\dot{\boldsymbol{y}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4}$$

with $\boldsymbol{J} = \partial \boldsymbol{k}/\partial \boldsymbol{q}$ the $n_y \times n_q$ motion task Jacobian matrix.

Relating the force task coordinates $\boldsymbol{f}$ to the trajectory in configuration space requires the consideration of the dynamic model of the robot in the Lagrangian form:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}. \tag{5}$$

Here, $\boldsymbol{B}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ collects velocity and gravitational terms, and $\boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}$ is the portion of the actuator torques $\boldsymbol{\tau}$ needed to balance the torques induced at the joints by the contact force $\boldsymbol{f}$. Note that we are implicitly assuming that the contact with the environment takes place at the same point (e.g., the end-effector) where the motion task coordinates $\boldsymbol{y}$ are defined. However, this assumption does not involve any loss of generality, because it may be easily removed by considering different Jacobian matrices in (4) and (5).

Then, the assigned motion-force task consists of a desired motion path $\boldsymbol{y}_d(s) \in \mathcal{Y}$ for the motion task variables $\boldsymbol{y}$ and a desired force path $\boldsymbol{f}_d(s) \in \mathcal{F}$ for the force task variables $\boldsymbol{f}$, where $s \in [s_i, s_f]$ is a path parameter. A typical choice for s is the arc length along the motion task path. Note that a desired force $\boldsymbol{f}_d(s)$ may only be specified for those values of s at which the motion task puts the robot in $\boldsymbol{y}_d(s)$ contact with the environment; this may happen throughout the task or intermittently, depending on the considered application. At those values of s where the robot is not in contact with the environment, we assume $\boldsymbol{f}(s) = \boldsymbol{0}$. In this case, eq. (5) indicates that the actuator torques are converted in pure motion. Finally, note that the the motion-force task can be specified directly as a trajectory; in this case, $s = t$.

2.3 Planning problem

At this point, we are ready to formulate our planning problem, which consists in finding a feasible configuration space trajectory (i.e., one over which joint, velocity and torque limits are satisfied) such that the assigned motion-force task is executed and collisions with obstacles are avoided. Clearly, for the planning problem to be well-posed, we must take the assumption that the robot is redundant with respect to the assigned motion-force task, i.e., $n_q > n_y + n_f$.

In the proposed framework, a solution to this problem is built as the composition of three parts: a configuration space path, representing the geometric part of the solution, a time history along the path, and a torque profile along the resulting trajectory.

More precisely, a solution consists of a path $\{\boldsymbol{q}(s) \in \mathcal{C},\ s \in [s_i, s_f]$, a *continuous* time history $s(t) : [0, T] \mapsto [s_i, s_f]$, and a torque profile $\boldsymbol{\tau}(t)$, $t \in [0, T]$, such that :

1. $s(0) = s_i$ and $s(T) = s_f$;

2. for all $t \in [0, T]$, it is $\boldsymbol{y}(t) = \boldsymbol{k}(\boldsymbol{q}(s(t))) = \boldsymbol{y}_d(s(t))$;

3. for all $t \in [0, T]$, it is $\boldsymbol{f}(t) = \boldsymbol{f}_d(s(t))$;

4. $\boldsymbol{q}(0) = \boldsymbol{q}_i$, $\dot{\boldsymbol{q}}(0) = \dot{\boldsymbol{q}}_i$, $\dot{\boldsymbol{q}}(T) = \dot{\boldsymbol{q}}_f$;

5. for all $t \in [0, T]$, it is $|\boldsymbol{q}(t)| \leq \boldsymbol{q}_M$, $|\dot{\boldsymbol{q}}(t)| \leq \dot{\boldsymbol{q}}_M$ and $|\tau(t)| \leq \tau_M$;

6. for all $t \in [0, T]$, it is $\mathcal{R}(\boldsymbol{q}(s(t))) \cap \mathcal{O}(t) = \emptyset$;

Requirement 1 simply state that the robot motion must start and stop on the endpoints of the task path $\boldsymbol{y}_d(s_i)$ and $\boldsymbol{y}_d(s_f)$ respectively. Together with continuity, this guarantees that all values of $s \in [s_i, s_f]$ are generated. Requirements 2 and 3 guarantee that motion-force tasks are satisfied at any configuration of the solution trajectory. Requirement 4 establishes that the robot matches the initial configuration and initial and final velocities (typically zero). Requirement 5 allows to satisfy joint position, velocity and torque bounds. Finally, last condition requires to avoid collisions with the obstacles (self-collisions can be tested as well).

Some further explanation is in order with respect to requirement 3 above. Differently from the motion task variables $\boldsymbol{y}(t)$ in requirement 2, the value of the force task variables $\boldsymbol{f}(t)$ at time t cannot be expressed as a pure function of the configuration $\boldsymbol{q}(t)$. Indeed, eq. (5) entails that $\boldsymbol{f}(t)$ results from the part of the actuator torques $\boldsymbol{\tau}$ that is not converted into motion; i.e., it is a function of $\boldsymbol{\tau}(t)$, $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$ and $\ddot{\boldsymbol{q}}(t)$. Since this function cannot be written in closed form, the planner will satisfy the force task constraint using a different mechanism than the motion task constraint. In particular, it will explicitly generate the joint torques that will drive the robot so as to realize the desired motion-force task. These joint torques will then be explicitly included in the solution.

Note the following facts.

- In the general case $s \neq t$, the time history $s(t)$ is not required to be non-decreasing: s must start at s_i and end at s_f, but it is not required to be monotonic along the trajectory. This means that, 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, whereas the actual motion $\boldsymbol{y}_d(t)$, $t \in [0, T]$, will depend on the choice of $s(t)$.
- The initial configuration $\boldsymbol{q}_i$ is assumed to be given. If this is not the case, a suitable $\boldsymbol{q}_i$ may be preliminarily computed by inverse kinematics.
- The final configuration $\boldsymbol{q}(s_f) = \boldsymbol{q}(T)$ and (if $s \neq t$) the total duration T of the motion are not assigned, and will be generated by the planner.

3 The case of pure motion tasks

To fully understand the structure of our planner for the general case of motion-force tasks, it is convenient to start from the simple case of a pure motion task. This will allow us to introduce the basic tools that will also be used in the general case.

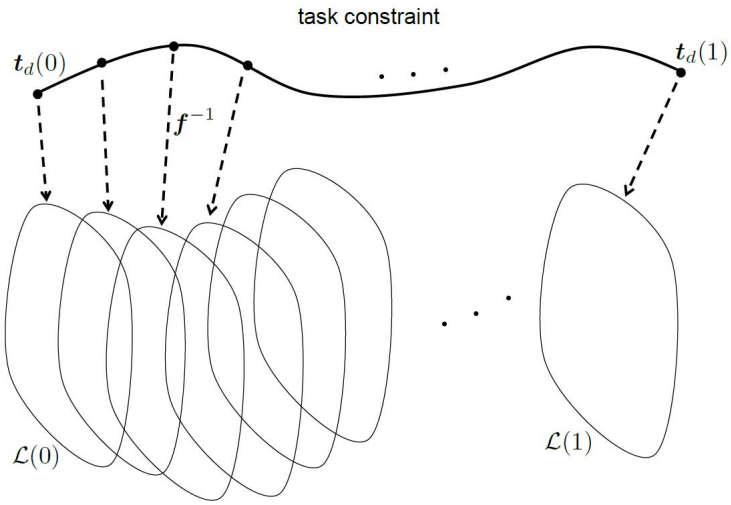


Figure 1: Leaves of the task-constrained configuration space $\mathcal{C}_{\text{task}}$, decomposed as a foliation.

3.1 Task-Constrained Motion Planning (TCMP)

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

Our solution to the TCMP problem relies on the principle of control-based motion planning [1], a paradigm where configuration samples are generated using a differential model of the robot (called motion generation scheme in the following). In particular, in [2] a motion generation scheme was introduced that is able to guarantee continued satisfaction of the constraints. The use of this scheme leads to a sampling-based randomized planner that achieves accurate execution of the task without increasing the size of the roadmap.

3.2 TCMP problem formulation

With reference to the previous problem formulation for the general case, assume now that the assigned task consists only of a motion task. That is, a desired path is assigned for the task variables $\boldsymbol{y}$ in the form $\boldsymbol{y}_d(\sigma)$, with $\sigma \in [0,1]$, w.l.o.g, and that $\boldsymbol{y}_d(\sigma)$ is differentiable. For the problem to be well-posed, we assume that:

$$\boldsymbol{y}_d(\sigma) \in \mathcal{T}^* \quad \forall \sigma \in [0,1], \tag{6}$$

where $\mathcal{T}^* \subset \mathcal{T}$ is the *non-singular task space*, defined as the set of regular and co-regular task space points. The workspace $\mathcal{W}$ is assumed to be populated by fixed obstacles only.

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

[1] We use different parameterizations for $\boldsymbol{y}_d$ and $\boldsymbol{q}$ to take advantage of the possibility of performing self-motions or backward motions.

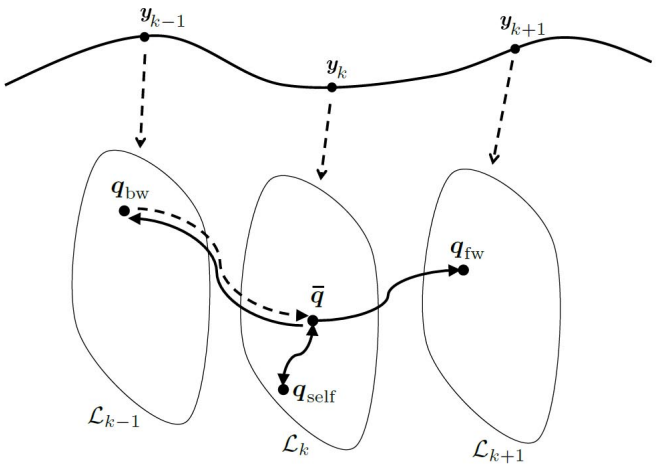


Figure 2: Possible extensions of the tree from the node $\bar{\boldsymbol{q}}$.

1. $\boldsymbol{y}(s) = \boldsymbol{k}(\boldsymbol{q}(s)) = \boldsymbol{y}_d(\sigma(s)), \forall s \in [s_i, s_f]$ (see equation (3));
2. the robot does not collide with obstacles or with itself.

The planning space for the TCMP problem is

$$\mathcal{C}_{\text{task}} = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{y}_d(\sigma) \text{ for some } \sigma \in [0,1]\}. \tag{7}$$

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

$$\mathcal{L}(\sigma) = \{\boldsymbol{q} \in \mathcal{C} : \boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{y}_d(\sigma)\}. \tag{8}$$

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

Our control-based randomized planner builds a Rapidly exploring Random Tree (RRT, see [3], [4]) in the task-constrained configuration space $\mathcal{C}_{\text{task}}$ to search for a collision-free path. For the construction of the tree, we make use of samples of the desired task path $\boldsymbol{y}_d(\sigma)$; in particular, denoting by $\{\sigma_1 = 0, \sigma_2, \ldots, \sigma_{N-1}, \sigma_N = 1\}$ an equispaced sequence of N path parameter values, let $\boldsymbol{y}_k = \boldsymbol{y}_d(k)$ (we drop the d subscript for compactness). The tree edges are collision-free subpaths obtained by applying the following motion generation scheme:

$$\boldsymbol{q}' = \tilde{\boldsymbol{v}} \tag{9}$$
$$\tilde{\boldsymbol{v}} = \boldsymbol{J}^{\dagger}(\boldsymbol{q})(\boldsymbol{y}'_d + k_t\, \boldsymbol{e}_t) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q}))\tilde{\boldsymbol{w}}, \tag{10}$$

where $\boldsymbol{J}^{\dagger}$ is the pseudoinverse of the task Jacobian $\boldsymbol{J}$, k_t is a positive gain, $\boldsymbol{e}_t = \boldsymbol{y}_d - \boldsymbol{y}$ is the task error, $\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ is the orthogonal projection matrix in the null space of $\boldsymbol{J}$, and $\tilde{\boldsymbol{w}}$ 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. 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 $\tilde{\boldsymbol{w}}$.

[2]Since this is a planning scheme, stability is required for reducing the drift associated to a numerical integration of (9–10).

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 (9–10) 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. Taking advantage of the different parameterizations of the task space and the configuration space paths, one may also perform a motion on the same leaf (self-motion) or even move backwards from the current leaf towards previous leaves. In formulas:

$$\boldsymbol{y}'_d = \frac{d\boldsymbol{y}_d}{ds} = \frac{d\boldsymbol{y}_d}{d\sigma} \cdot \frac{d\sigma}{ds} \tag{11}$$

where, for $d\sigma/ds = 1/0/-1$ we obtain, respectively, forward, self- and backward motions. Figure 2 shows possible expansions of the tree in correspondence of the choices for ds/ds. The tree is extended from $\bar{\boldsymbol{q}}$ using the motion generation scheme (9–10) to perform:

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

Whenever a singular configuration is generated during FM, BM or SM, the integration stops. Besides, for each generated subpath, a procedure is called to verify that the terminal configuration ($\boldsymbol{q}_{\text{fw}}$, $\boldsymbol{q}_{\text{self}}$ or $\boldsymbol{q}_{\text{bw}}$) is neither singular nor in collision with the obstacles.

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

3.3 Extensions: cyclicity and moving obstacles

We now discuss two relevant extensions of the basic TCMP problem.

Cyclicity

Traditionally, task-constrained motion in configuration space for redundant robots is generated through kinematic control techniques [5], [6], [7], [8]. 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.

With reference to the case of pure generalized inversion (no null-space motions), Shamir and Yomdin [9] 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 [10], was exploited to design cyclic generalized inverses, e.g., in [11] and to achieve asymptotically cyclic kinematic control in [12]. Other works on cyclicity include, e.g., [13], [14], [15]. 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 [16], [17]. This

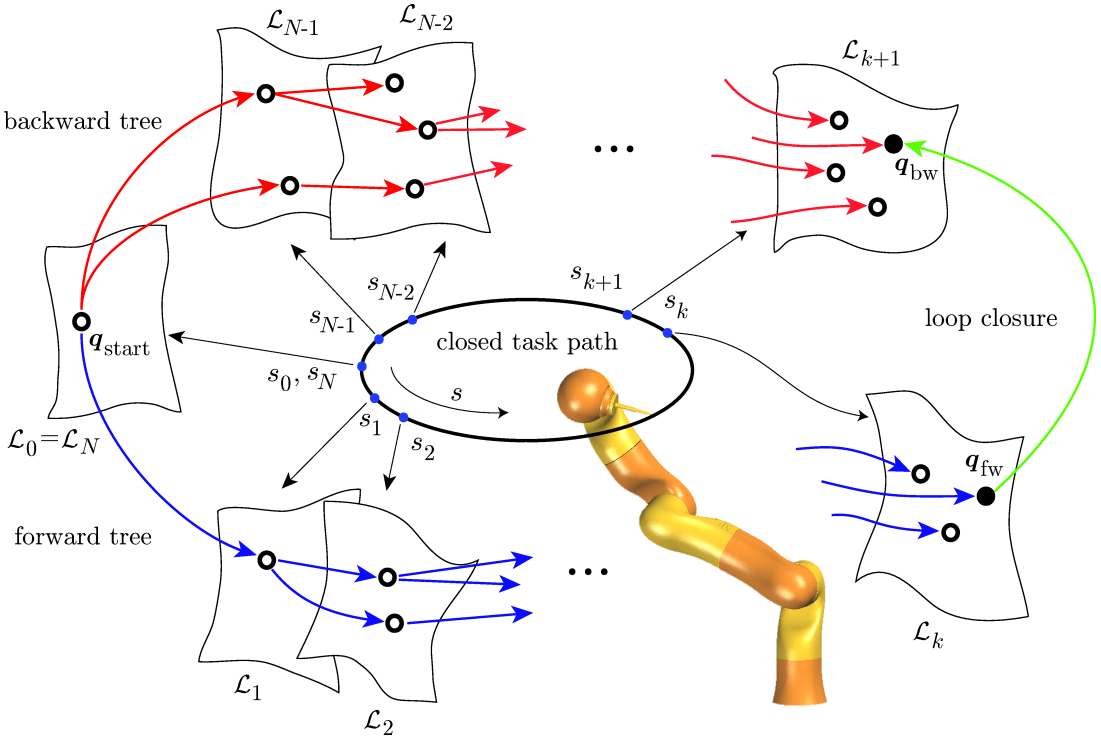


Figure 3: The C-TCMP (Cyclic-TCMP) planner relies on bidirectional search and loop closure in the task-constrained configuration space.

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 [18] we presented a solution for the extension of the TCMP problem to the cases of cyclic motion tasks in order to preserve cyclicity in the joint space. In particular, 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 approach to the solution of the C-TCMP problem that we presented in [18] is illustrated in Figure 3. Since the problem addressed is a special case of Task-Constrained Motion Planning (TCMP), we adopt the terminology and framework introduced in [2]. In summary, a bidirectional search of the task-constrained configuration space $\mathcal{C}_{\text{task}}$ is performed by growing two Rapidly-exploring Random Trees. 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. Trees extension is obtained integrating the motion generation scheme (9–10). 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 $\mathcal{L}_0 = \mathcal{L}_N$.

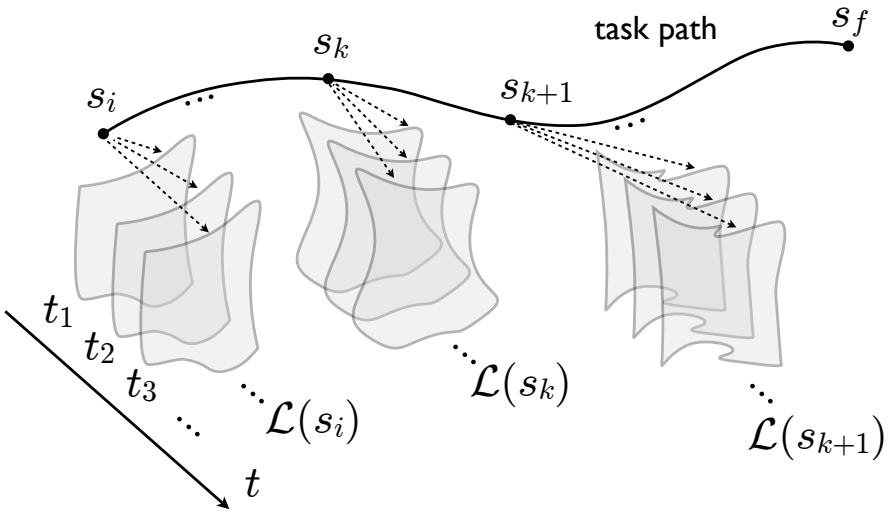


Figure 4: The structure of $\mathcal{S}_{\text{task}}$: each leaf $\mathcal{L}(s)$ is the set of configurations satisfying the task constraint at s replicated along the time axis.

Moving obstacles

Another relevant extension of the basic formulation of the TCMP problem, which is reflected in many real world applications, considers the presence of obstacles that move along trajectories, whose predictability can range from fully known to completely unknown. In [19] we proposed a new planning algorithm to address the TCMP problem in presence of motion task constraints and 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 [20]. Early solutions (like, e.g., [21], [22], [23]) 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 [24], [25], [26].

All the above methods prove to be prohibitively inefficient when dealing with articulated robotic systems, due to the computational complexity of the problem.

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 [27] or RRT [4], 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 [28].

Our control-based approach proposed in [19] is conceptually different. It guarantees a continuous satisfaction of the task constraint and probabilistic completeness. With respect to projection-based methods, and also to planners the control based planner allows to arbitrarily improve the task accuracy without increasing the roadmap complexity. We called this extension to basic TCMP problem Task Constrained Motion Planning with Moving Obstacles, or TCMP_MO. In [19] as an intermediate position with respect to realistic problems we assumed that obstacles move along trajectories that are known in advance.

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

$$\mathcal{S}_{\text{task}} = \{(\boldsymbol{q}, t) \in \mathcal{S} : \boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{y}_d(s), s \in [s_i, s_f]\}.$$

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

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

On each leaf, t can assume any value in $[0, \infty)$. Figure 4 illustrates the structure of $\mathcal{S}_{\text{task}}$: the set of configurations satisfying the task constraint at s (the so-called *self-motion manifold* associated to the task value at s) is replicated along the time axis to form a leaf of $\mathcal{S}_{\text{task}}$.

4 Extending to general tasks

4.1 Background

In the real world, robotic systems are usually assigned tasks that are more general than pure motion tasks. A typical robotic application, both in industrial and in service domains, involves contacts and interactions with humans and/or with the environment. For such applications, the motion planning problem must be described considering the presence of hybrid motion-force tasks.

Consider for instance, the problem of planning the movement of a robotic system involved in welding or cutting assignments in industrial applications. Both these type of operations require to specify a desired task path for the end-effector as well as a desired force profile along the points of the task path at which the end-effector is in contact with the environment (i.e. the object on which the operation must take place).

Hybrid motion-force tasks have been most often considered in literature from the control point of view, while only a few works have been proposed in the past to handle force or hybrid motion-force tasks in motion planning problems. Most of them were introduced in the context of grasp planning and robotic walking. In [29] the authors propose methods for solving the grasp manipulation problem in presence of contact forces and actuators bounds. In [30] and [31] some methods for finding smooth actuator efforts along pre-defined joint motions have been proposed. In [32] a method called *Force-Workspace* is proposed to generate motions for a mobile robotic system to handle a given set of loads over long motions satisfying the actuators bounds without violating contact constraints and the kinematic constraints intrinsic of the structure. This method plans feasible motions in the configuration space where the force task and the other constraints are mapped to form constraint obstacles, similar to geometric C-space obstacles.

All these methods provide important tools, but, at the best of our knowledge, the path planning problem in presence of hybrid motion-force tasks and kinematic and dynamic constraints with moving obstacles has not been addressed yet.

We propose a control-based RRT-like path planning algorithm to solve this complex problem, as evolution of the framework developed for the TCMP problem with pure motion tasks and the subsequent extensions introduced in Section 3.3, concerning the cases of cyclic tasks and moving obstacles. The proposed solution is based on a second-order version of the motion generation scheme introduced in Section 3.2 (eq.(9–10)).

4.2 Motion generation

Referring to the notation introduced in Section 2, we define 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, the motion task 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 motion task. In formulas, the planning space is:

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

From a geometric viewpoint, $\mathcal{S}_{\text{task}}$ is a manifold with boundary which foliates in *leafs*:

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

with each leaf associated to a value of parameter $s \in [s_i, s_f]$

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

Figure 5 illustrates the structure of the leaves.

The existence of a solution to the planning problem depends on the interplay between the desired task 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 the bounds on the available velocities and torques are considered.

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 occur 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 $\mathcal{L}_k$. While the same value of $s = s_k$ is shared by all vertexes on $\mathcal{L}_k$, a different value of $\dot{s} = \dot{s}_V$ is associated to each vertex, as a byproduct of the subtrajectory that generated that vertex.

The generalized velocities and accelerations associated to a particular solution can be written as

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{q}'(s)\dot{s}(t) \tag{12}$$

and

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{q}''(s)\dot{s}^2(t) + \boldsymbol{q}'(s)\ddot{s}(t), \tag{13}$$

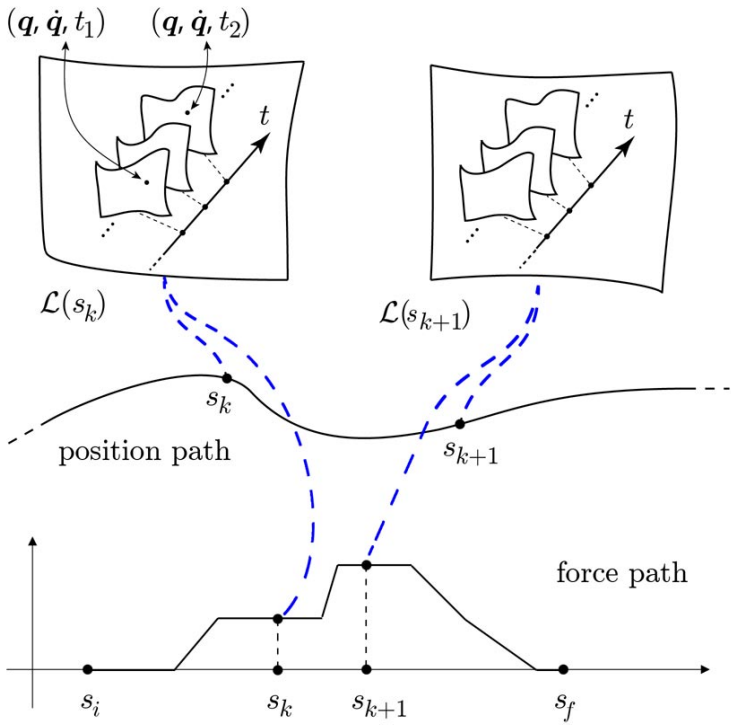


Figure 5: $\mathcal{S}_{\text{task}}$ is a foliation: each leaf $\mathcal{L}(s)$ is the set of points $(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \in \mathcal{S}$ such that $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are consistent with the motion task constraint for a certain value of s, while t may assume any value.

where we have used the notation $()' = d()/ds$. Hence, 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

$$\ddot{s} = \ddot{s}_V, \tag{14}$$

with $\ddot{s}_V$ a constant value chosen within a predefined range $[-c_{\text{max}}, c_{\text{max}}]$. 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 kind 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}.

Geometric accelerations are generated using the second order motion generation scheme

$$\boldsymbol{q}''_V(s) = \boldsymbol{J}^\dagger(\boldsymbol{y}''_d - \boldsymbol{J}'\boldsymbol{q}' + \boldsymbol{K}_p\boldsymbol{e}_y + \boldsymbol{K}_d\boldsymbol{e}'_y) + (\boldsymbol{I} - \boldsymbol{J}^\dagger\boldsymbol{J})\boldsymbol{a}_V \tag{15}$$

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 arbitrarily chosen n_q-dimensional vector which produces internal motions without affecting 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.

Using system state information corresponding to the vertex V (in particular s_V, $\dot{s}_V$, $\boldsymbol{q}_V$ and $\boldsymbol{q}'_V$), together with the value $\ddot{s}_V$ chosen in (14) and the value $\boldsymbol{q}''_V$ computed in (15), we are able to evaluate $\boldsymbol{q}_V, \dot{\boldsymbol{q}}_V, \ddot{\boldsymbol{q}}_V$ (by means of equations (12) and (13)) and consequently, using the dynamic model (5), the torques required to

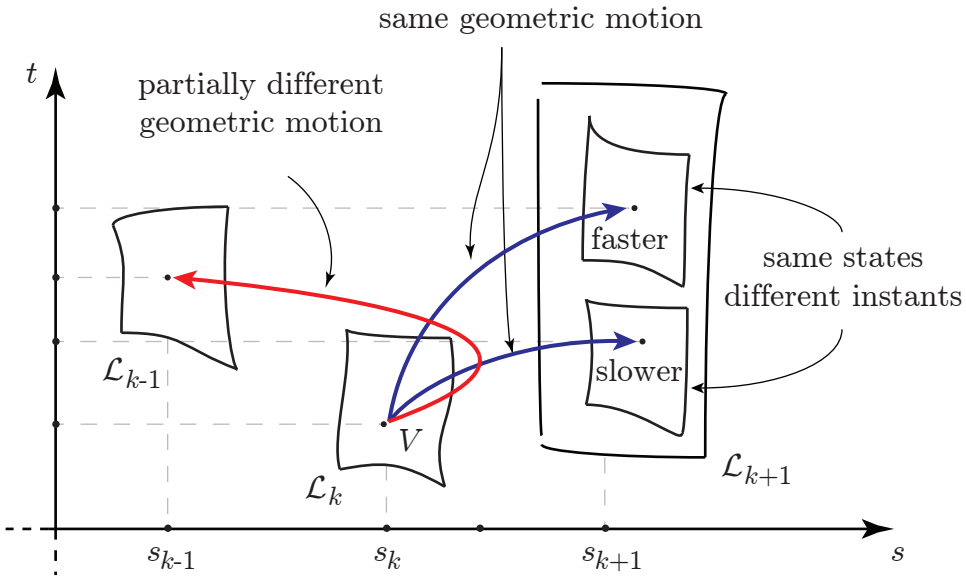


Figure 6: An illustration of motion generation within our planner. In this particular case, it is assumed that $\dot{s}_k > 0$. The two monotonic forward motions (blue) correspond to the same choice of $\boldsymbol{a}_V$ (hence, of $\boldsymbol{q}''_V$) but different positive values of $\ddot{s}_V$. The non-monotonic motion (red) is generated again by the same choice of $\boldsymbol{a}_V$ but now with a negative $\ddot{s}_V$. The geometric motion is the same of the forward case until the direction its reversed.

move the robot from the vertex V along the planned subtrajectory:

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

To this torque component, we can finally add the component that will guarantee execution of the force task:

$$\boldsymbol{\tau}_f = \boldsymbol{J}^T(\boldsymbol{q}_V)\boldsymbol{f}_d(s(\boldsymbol{q}_V)).$$

Finally, the total required torque for executing the hybrid task from V is obtained as:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_m + \boldsymbol{\tau}_f. \tag{16}$$

Equation (16) is used during the integration to test if the joint motions are dynamically feasible. Also velocity constraints are continuously verified together with avoidance of moving obstacles. If any of these is violated, the considered subtrajectory is discarded. Otherwise, integration proceeds until the subtrajectory lands on an adjacent leaf to $\mathcal{L}_k$, be it $\mathcal{L}_{k+1}$ or $\mathcal{L}_{k-1}$.

Figure 6 illustrates some typical situations encountered when applying the motion generation scheme from a vertex in $\mathcal{L}_k$.

4.3 The complete 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 motion-force tasks, 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 5).

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, obtained by applying the previous motion generation scheme. The planning tree is expanded using an RRT-like mechanism.

5 Smooth transitions between consecutive tasks

A peculiar feature of this kind of generalized TCMP problem is that the hybrid task must be appropriately assigned for the problem to be solvable, in the sense that some smoothness conditions must be met at the *transition points*, i.e., the points where the robot end-effector makes or leaves contact with the environment.

In particular, building upon the works initiated by Huang and McClamroch in [33], one may show that the motion and force tasks must satisfy the following conditions at each transition point:

1. the desired force $\boldsymbol{f}_d$ must be zero at the point;
2. if a non-zero entrance/exit speed is desired at the transition, the tangent to the motion path at the point should lie in the local tangent plane to the environment surface.

These conditions guarantee that no impact force is generated when contact with the surface is established, and that the generated robot commands will be continuous at the task transitions.

On the basis of the these conditions, smooth transitions can occur even with non-zero transition velocities, provided that the tangency requirement is met. In any case, the desired force must be zero at the transition. Figure 8 shows an example of a hybrid task with a smooth transition.

On the other hand, nonsmooth paths can be accepted as long as zero velocities are assigned at transition points. Figure 7 illustrates a situation in which, in order to avoid impact forces, the robot end-effector has to stop when making and leaving contact with the environment surface. Note that in this case, since the robot has zero velocity at the transition points, the profile of the force task is not required to be continuous there.

6 Planning experiments

The proposed 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 produced by Kineo CAM (Siemens). In this section, we report two planning experiments for a scenario involving a 3-dimensional positioning task and 1-dimensional force task for the tip of a KUKA LWR-IV 7-DOF manipulator mounted on a table. The degree of redundancy for this kind of task is 2 (the wrist roll is frozen as it does not contribute to tip positioning nor to the tip applied forces). Joint position, velocity and torque limits for this robot were taken from the official documentation. In this scenario, all obstacles are fixed and collisions with them as well as self-collisions must be always avoided.

We used the same settings in both experiments. In particular, the planner uses a sequence of $N = 11$ equispaced samples from the desired task path (including the endpoints, which correspond to $s = 0$ and $s = 1$). In the motion generation scheme, we use $\boldsymbol{K}_p = 300\, \boldsymbol{K}_d = 50 \cdot \boldsymbol{I}$, and the null space term is constrained to be in norm at most 100% of the range space term. The upper bound for $|\ddot{s}|$ is $c_{\max} = 1$. Integration is performed numerically using Euler method with step size $\Delta s = 0.002$.

In both experiments, the manipulator move its tip along a planar path contained in a horizontal plane while avoiding collisions with the table, the wall, the whiteboard and the lamp. Part of the motion path is in the free space and part lies on the whiteboard. In correspondence of the part of the motion task that is in contact with the whiteboard, a force task normal to the whiteboard is also specified.

In the first experiment, we assume a task with smooth transition from free space motion to motion in contact (see Figure 9). The points on the motion task corresponding to the values s_1 and s_2 of the parameter s

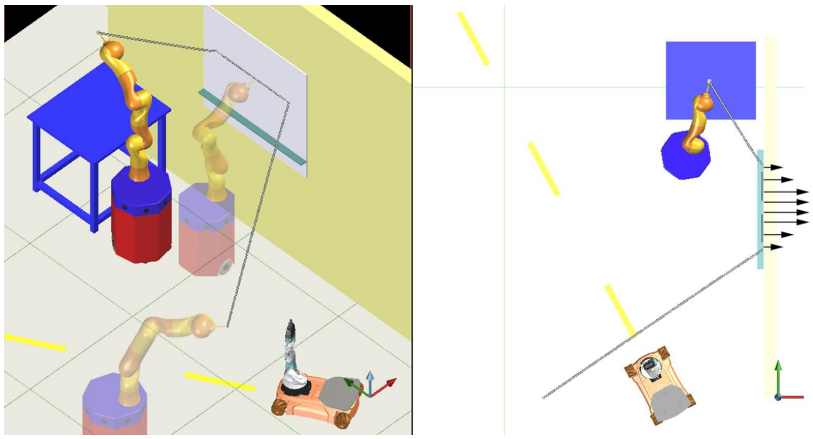


Figure 7: Non-smooth transition from free space motion task to hybrid motion-force contact task.

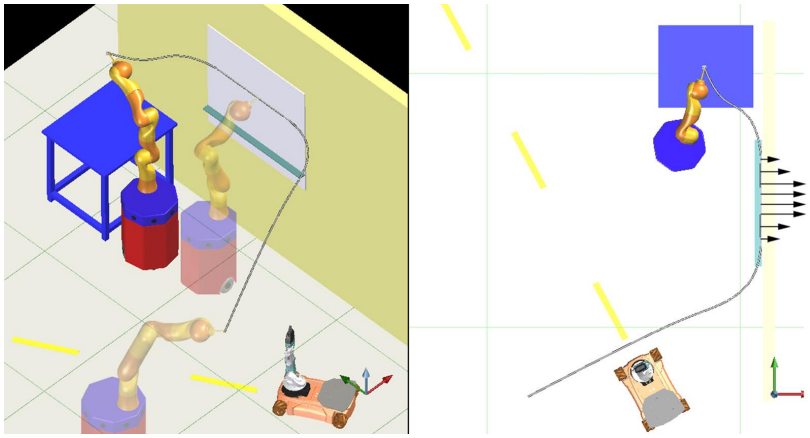


Figure 8: Smooth transition from free space motion task to hybrid motion-force contact task.

are the transition points. At those points, the desired force is zero and the tangent to the motion task path lies on the plane of the constraint. This scenario matches the conditions discussed in Section 5, and in particular a nonzero velocity on the transition points is admissible. Figure 10 shows some snapshots of a feasible solution computed by the proposed planner for this case. Figure 11 shows the torques required to accomplish the desired tasks (note that they are always within the bounds). Finally, the joint velocities are shown in Figure 12: note that they are zero at the beginning and at the end of the motion task, but nonzero at the transition points. Figure 13 shows the time history along the solution.

In the second experiment, we consider a motion-force tasks with nonsmooth transition from free space motion to motion in contact (see Figure 14). In this case, for the reasons explained in Section 5, a zero velocity must be assigned to the end-effector at transition points to avoid undesired impact forces. Using the proposed planner, a solution can be found as the union of the solutions of three separate subproblems, obtained by dividing the original motion-force task into three parts, i.e., $s \in [0, s_1]$, $s \in [s_1, s_2]$ and $s \in [s_2, 1]$ respectively. Figure 15 shows some snapshots from a feasible solution. Due to the fact that at the transition points the robot (not only the end-effector) stops, the time needed to complete the whole task in this case is quite larger than in the first case. Figures 16 and 17 show, respectively, the torques and the joint velocities along the solution. Note that the joint velocities are zero at the transition points.

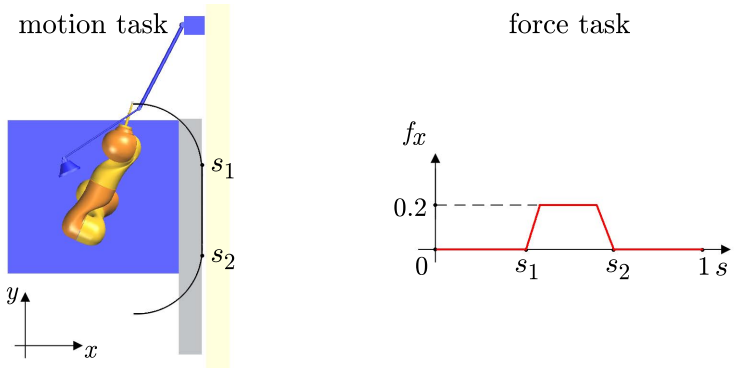


Figure 9: Experiment 1: Smooth transition from free space motion task to hybrid motion-force task.

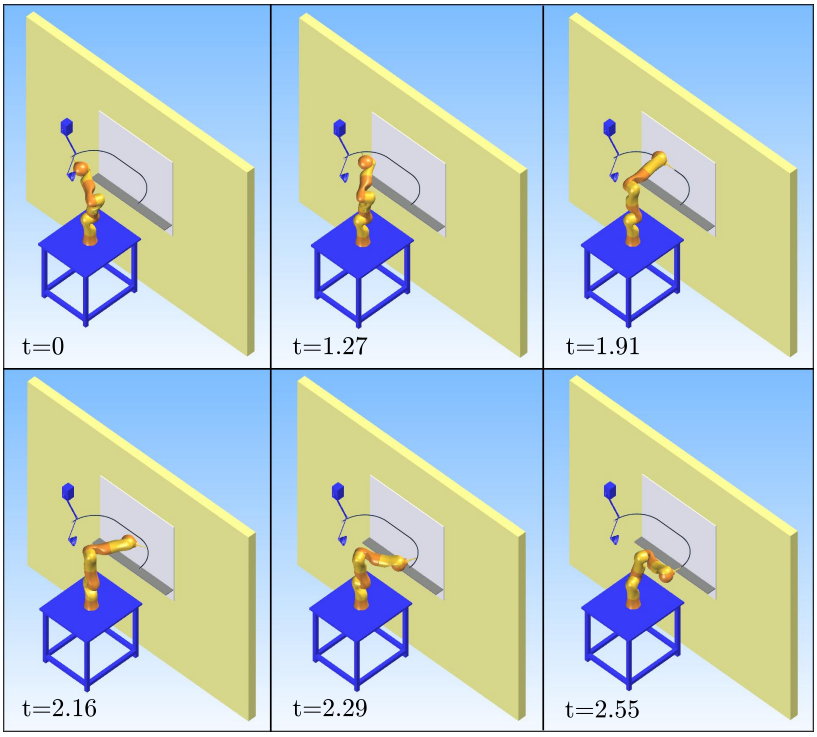


Figure 10: Experiment 1: Sample frames from the solution.

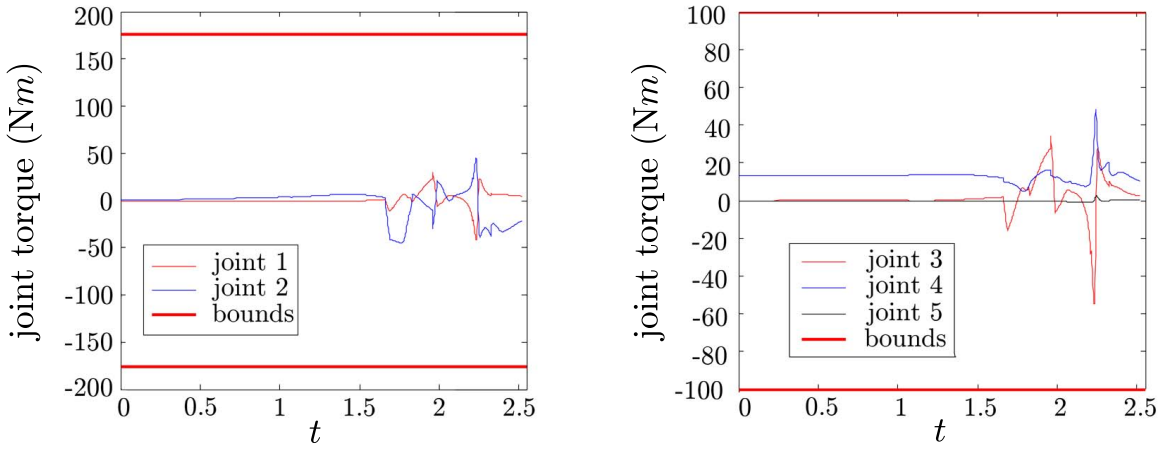


Figure 11: Experiment 1: Required torques.

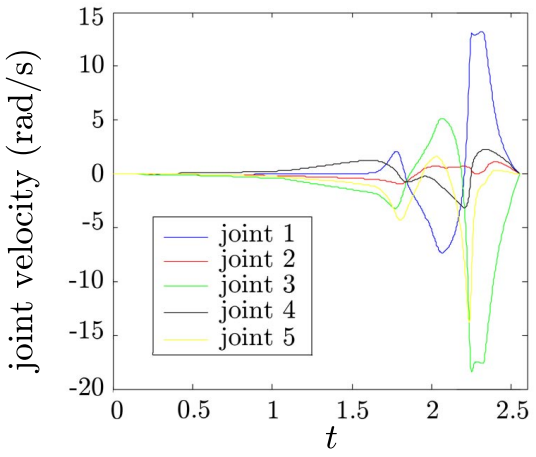


Figure 12: Experiment 1: Joint velocities.

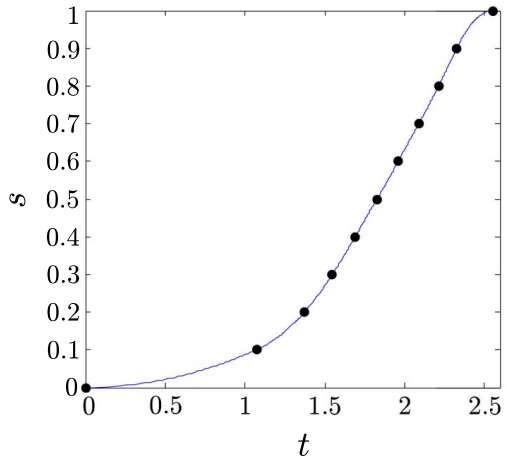


Figure 13: Experiment 1: Time history.

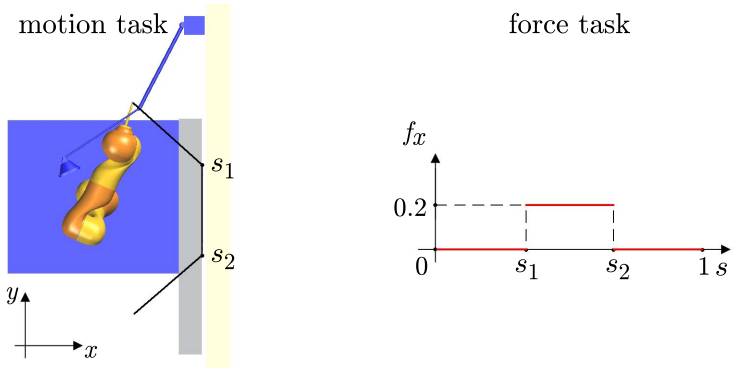


Figure 14: Experiment 2: Non-smooth transition from free space motion task to hybrid motion-force task.

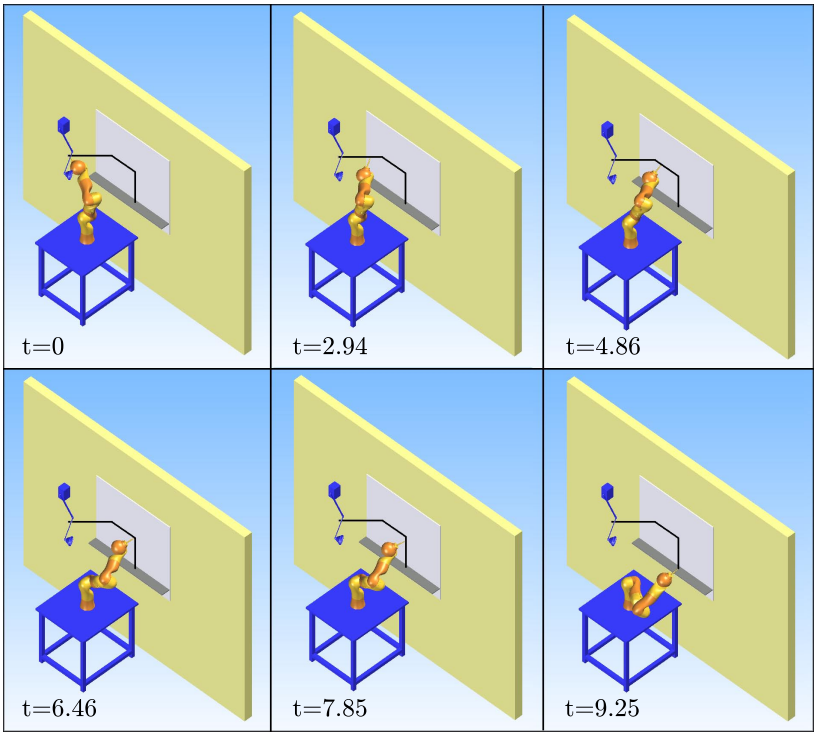


Figure 15: Experiment 2: Sample frames from the solution.

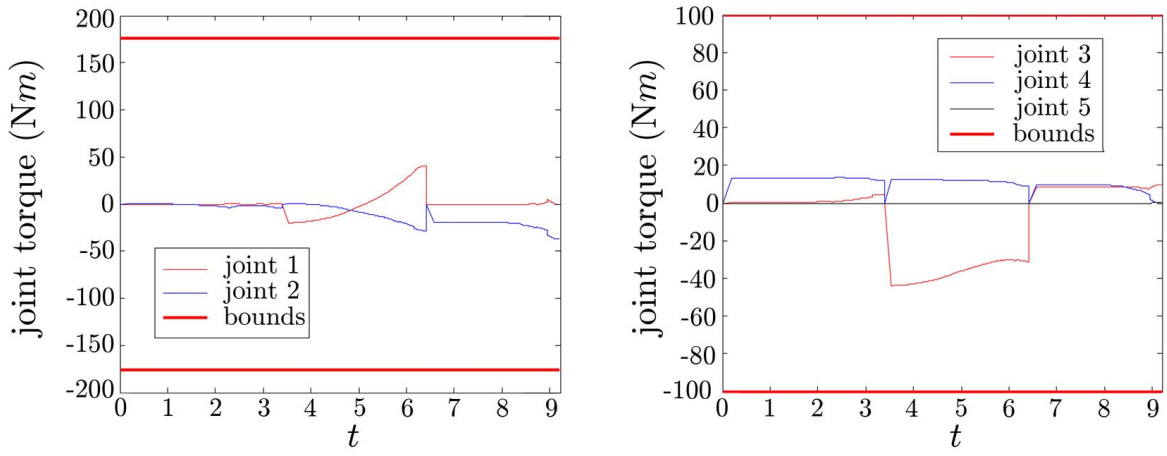


Figure 16: Experiment 2: Required torques.

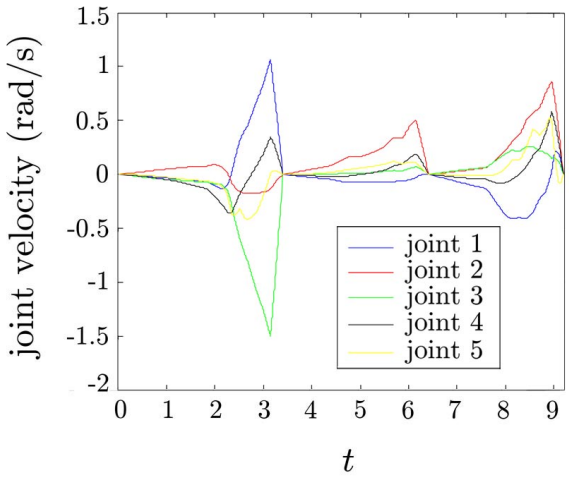


Figure 17: Experiment 2: Joint velocities.

References

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

[2] 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.

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

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

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

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

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

[8] S. Chiaverini and 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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*, 2013.

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

[21] 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.

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

[23] 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.

[24] 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.

[25] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *Int. J. of Robotics Research*, vol. 17, pp. 760–772, 1998.

[26] 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.

[27] 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.

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

[29] J. Kerr and B. Roth, “Analysis of Multifingered Hands,” *International Journal of Robotics Research* , vol.4, no.4, Winter 1986, pp. 3-17.

[30] C.A. Klein and S. Kittivatcharapong, “Optimal Force Distribution for the Legs of a Walking Machine with Friction Cone Constraints,” *IEEE Trans. on Robotics and Auto.*, vol 6, no. 1, Feb. 1990, pp 73-85.

[31] M.A. Nahon and J. Angeles, “Optimization of Dynamic Forces in Mechanical Hands,” *ASME Journal of Mechanical Design*, vol.113, June 1991, pp. 167-173.

[32] A. Madhani and S. Dubowsky, “Planning motions of robotic systems subject to force and friction constraints with an application to a robotic climber,” *Proceedings of the Ninth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, vol.187, 1993, pp. 129–139.

[33] H.P. Huang and N.H. McClamroch, “Time-optimal control for a robotic contour following problem,” *IEEE J. of Robotics and Automation*, vol.4, April 1988, pp. 140–149.