The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Optimization of Tasks Warping and Scheduling for Smooth Sequencing of Robotic Actions

François Keith[1,4], Nicolas Mansard[2], Sylvain Miossec[3], and Abderrahmane Kheddar[1,4]

[1]CNRS-UM2 LIRMM, Montpellier, France
[2]CNRS-LAAS, Toulouse, France
[3]PRISME-Univ. d'Orléans, Bourges, France
[4]CNRS-AIST JRL, UMI3218/CRT, Tsukuba, Japan
{*keith, kheddar*}*@lirmm.fr, nmansard@laas.fr, sylvain.miossec@bourges.univ-orleans.fr*

***Abstract*— This paper presents a method for sequencing a set of robotic tasks in an optimal way. Tasks description and execution are based on the task-function approach, which enables to build complex whole-body behaviors from local control laws. A naive solution to this problem would be to schedule the execution of the tasks sequentially, avoiding concurrency. This solution does not exploit full robot capabilities such as redundancy and have poor performance in terms of execution time or energy. However, reasoning on concurrent tasks is difficult while accounting for all the physical constraints of the robot. Our contribution is to determine the time-optimal realization of the mission taking into account robotic constraints that may be as complex as collision avoidance. Our approach achieves more than a simple scheduling; its originality lies in maintaining the task approach in the formulated optimization of the task sequencing problem. This theory is exemplified through a complete experiment on the real HRP-2 robot.**

I. Introduction

A robot is designed to perform missions in various application contexts. When the environment is well or partially structured most missions can be hierarchically decomposed. That is, missions undergo functional objective decomposition into a set of processes or operations that can be defined as templates. Each operation can be decomposed into a set of tasks (i.e. generic sensory-motor functions), and each task can be easily mapped into robot execution. The whole scheme may constitute an exploitable generic skill/behavior. Yet, various levels of decomposition can be achieved depending on the envisaged software/hardware implementation, additional environment constraints, the human-machine interface, etc. In the end, the robot is assigned with a sequence of tasks to realize a given mission.

Numerous works have been proposed to compute such a sequence of tasks from a given mission and a set of causal paradigms [2], [5]. However, they generally produce a symbolic plan, where the only numerical precisions lie on the scheduled time data. Its robotic application into the real world requires the time sequence to be refined, typically through an applicative path planner [9], that will compute the trajectories to be followed by the robot. Yet, the meaning of the symbolic plan is lost in the global trajectory. Such low-level methods lack of robustness to environment changes or uncertainties. Consequently, the remaining trajectory may have to be recomputed several times while the mission is being achieved. Moreover, it is difficult (and then often specifically hard coded) to enhance the numerical trajectory with symbolic data, that would help re-computing only part of the plan [16] or distort locally the trajectory to apprehend small environment changes [17], [8].

Rather than using a trajectory planner between the temporal reasoning and its real robotic execution, we propose to use a sensory-motor control approach based on task components. The task function [18] or the operational space formulations [7] are elegant approaches to produce intuitively robot objectives. They also allow to address the control problem directly in the sensor space, improving robustness of the action execution against environment uncertainties and variability [3]. They are trajectory free, which means that it is not necessary to explicitly compute all trajectories before the execution or during the execution, namely in response to environment changes. Moreover, since a same task space is valid for a large set of robots, a control scheme based on task formalism is certainly portable and easy to modify and to maintain. In addition, these methods include a kinematics or dynamic formulation that decouples the task space from the null-space (*i.e.* the joint space that let the task invariant) [11], [6]. A secondary task can then be applied in the null-space, and, recursively, a hierarchic set of tasks (or *stack of tasks*, or SoT) can be considered [15], [21]. Hierarchy of tasks are becoming popular to build complex behavior for very redundant robot such as humanoids [1], [13], [20], [19]. The formalism introduced in [13] proved to be efficient in dealing with complex humanoid missions: the SoT is mainly a hierarchy of tasks driving the robot while ensuring locally a given set of constraints to be satisfied. We make use of this formalism (Section II).

A task (*i.e.* a *task function* [18]) can be directly linked to the symbols on which the task temporal network is reasoning (*e.g.* reaching an object to be grasped is a task that requires the robot arm to be available, and whose post-condition is to have the gripper on the object – it is also directly described by a sensory-motor function applicable to the SoT).

Mission decomposition is thus executable directly by a SoT, which guarantees good robustness and avoid unnecessary trajectory (re)computation. However, exclusive task sequencing on the robot produces generally jerky movements

which may look to humans as monotonous automated motions. On the other hand, it is difficult for the temporal network to produce a scheduling with task overlapping when the tasks concurrency is restricted by physical limitations of the robot (for example obstacles or dynamical constraints on a humanoid). Since the problem is not in a standard discreet form, symbol-based task scheduling techniques can not apply straightforwardly. On the other hand, semi-infinite optimization techniques [10][14] have been used to generate low level trajectories for the overall execution, while accounting for such constraints. Such trajectory-based approaches are generally insufficiently robust to environment uncertainties.

In this paper, we propose to rely on task for both the overall symbolic reasoning and the control on the robot. In between, we propose to use semi-infinite optimization to refine the symbolic schedule and account for (numeric) robotic constraints. Given a set of elementary tasks sequence to achieve a given mission, our solution returns for each task, the optimal times at which it is put and removed from the SoT and also the optimal parameters for the task execution (*e.g.* control gain). We additionally expect from this method a *smooth tasks sequencing* (i.e. smooth transitions of tasks through task overlapping). The originality of our approach lies in keeping the task component in the optimization formulation of this problem, which can roughly translate to optimizing tasks overlapping by manipulating tasks, i.e. the controllers, as *variables* of the optimization problem. The task formulation details are first recalled in Section II. A generic solution for optimizing a task sequence is then detailed in Section III. The theory is finally exemplified through an experiment with a real HRP-2 robot, the mission consisting in getting a can from a fridge.

II. Generic Task Sequencing

A. Task function formalism and Stack of Tasks

Defining the motion of the robot in terms of task simply consists in choosing several control laws to be applied on a subpart of the robot degrees of freedom (DOF).

A task is defined by a vector $\mathbf{e}$ (typically, the error between a signal $\mathbf{s}$ and its desired value, $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$). The Jacobian of the task is noted $\mathbf{J} = \frac{\partial e}{\partial q}$, where $\mathbf{q}$ is the robot configuration vector. In the following, we consider that the robot input control is the joint velocity $\dot{\mathbf{q}}$. The equation of motion is thus reduced to the kinematics:

$$\dot{\mathbf{e}} = \mathbf{J}\dot{\mathbf{q}} \tag{1}$$

Considering a reference behavior $\dot{\mathbf{e}}^*$ to be applied in the task space, the control law to be applied on the robot whole body is given by the least-square solution:

$$\dot{\mathbf{q}} = \mathbf{J}^+\dot{\mathbf{e}}^* + \mathbf{P}\mathbf{z} \tag{2}$$

where $\mathbf{J}^+$ is the least-square inverse of $\mathbf{J}$, $\mathbf{P} = \mathbf{I} - \mathbf{J}^+\mathbf{J}$ is the null-space of $\mathbf{J}$ and $\mathbf{z}$ is any secondary criterion that will be applied without disturbing the main task thanks to the projection into $\mathbf{P}$[1]. A typical requested behavior is the regulation of the error, which can be obtained through an exponential decrease by setting:

$$\dot{\mathbf{e}}^* = -\lambda\mathbf{e} \tag{3}$$

As mentioned earlier, (2) enables to compose a complex behavior from a set of tasks [21], [1], [19]: $\mathbf{z}$ can be used to fulfil a secondary task, *without* disturbing the main task having priority. This nice decoupling can be extended recursively to a set of n tasks, each new task being fulfilled without disturbing the previous ones. The complete implementation of this approach is proposed in [12] under the name *Stack of Tasks* (SoT). The structure enables to easily add or remove a task, or to swap the priority order between two tasks, during the control. Constraints (such as joints limit) can be taken into account. The continuity of the control law is preserved even at the instant of change.

B. Gain handling

The simple attractor presented in (3) has the advantage to introduce a nice exponential decrease. However, it can be penalizing, since $\dot{\mathbf{q}}$ is directly proportional to $\mathbf{e}$ (3) . At the beginning of the task, $\|\mathbf{e}\|$ reaches its higher value (strong acceleration), while at the end of the task, $\|\mathbf{e}\|$ decreases slowly (slow convergence).

A very classical 'trick' when regulating a task is to rather use an adaptive gain $\lambda = \lambda(\mathbf{e}(t))$ that depends on the norm of the error of the task. To keep the nice property of the attraction, the gain only adapts with the error, and not directly with the time. We choose the following function:

$$\lambda(\mathbf{e}) = (\lambda^F - \lambda^I)\exp\left(\frac{-\|\mathbf{e}\|\beta}{\lambda^F - \lambda^I}\right) + \lambda^I \tag{4}$$

with λ^I the gain at infinity, λ^F the gain at regulation (such as $\lambda^I \leq \lambda^F$) and β the slope at regulation.

C. Sequence of tasks

A task sequence is a finite set of tasks sorted by order of realization, and eventually linked to each other. Any pair of tasks can be either independent (i.e. they can be achieved in parallel) or constrained (i.e. one may have to wait for another one to be achieved, so as to make sense or to be doable).

The sequence can be formulated into a classical temporal network scheduling, starting at t^0 and ending at t^{End}. Both values are finite and the sequence does not loop. Besides, we may consider for the sake of clarity but without loss of generality that each task appears only once in the sequence.

The *position* of a task in the sequence is defined by the time interval during which it is maintained in the SoT. For a given task i, this interval is noted $[t_i^I, t_i^F]$: the task enters in the SoT at t_i^I and is removed at t_i^F. These instants are defined with respect to the beginning of the sequence at t^0. However, they do not indicate the achievement level of the task: t_i^F may apply before the task regulation. Let's ϵ_i be the tolerance on the task regulation: a task is considered as regulated when $\|\mathbf{e_i}(t)\| \leq \epsilon_i$. The regulation time t_i^R is defined by $\|\mathbf{e_i}(t_i^R)\| = \epsilon_i$.

[1]Eq. (2) is the least-square solution when $\mathbf{z} = \mathbf{0}$

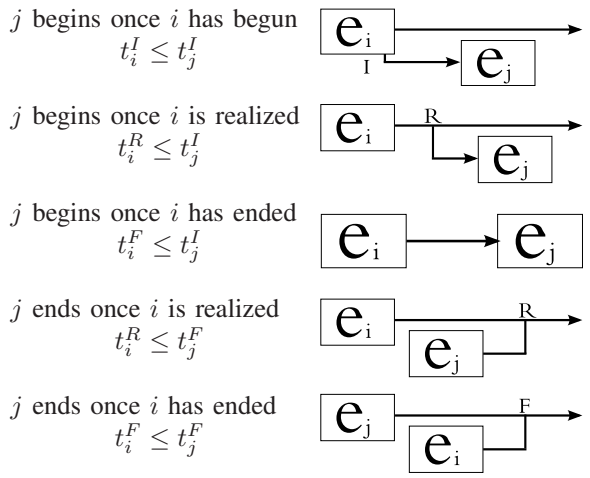

Fig. 1. Five time-dependency relations are considered.

A task sequence is characterized by a set of time-constraints binding the schedules of two tasks $\mathbf{e_i}$ and $\mathbf{e_j}$. They can be defined as follow[2]: $\mathbf{e_i}$ must begin or end once $\mathbf{e_j}$ has begun, has ended or has been regulated.

We use the graphical representation given by Fig. 1 and the following notation to describe the sets of pairs of tasks $\mathbf{e_i}$ and $\mathbf{e_j}$ that undergo these dependencies ($\mathbf{e_i}$ is the direct predecessor of $\mathbf{e_j}$) :

$$S_{I,I} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^I \leq t_j^I\} \tag{5a}$$
$$S_{R,I} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^R \leq t_j^I\} \tag{5b}$$
$$S_{F,I} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^F \leq t_j^I\} \tag{5c}$$
$$S_{R,F} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^R \leq t_j^F\} \tag{5d}$$
$$S_{F,F} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^F \leq t_j^F\} \tag{5e}$$

For example, the robot has first to grasp an object and maintain the force closure on it ($\mathbf{e_A}$) before moving it ($\mathbf{e_B}$). The task ($\mathbf{e_B}$) can only start once the task ($\mathbf{e_A}$) has been realized, and must end before the task ($\mathbf{e_A}$).

III. Continuous optimization of sequence of tasks

Given a set of hypothesis described using (5), we now propose a generic solution to automatically compute an optimal set of task-behavior parameters and their sequencing plan to be executed by the SoT.

A. General problem formulation

An optimization problem is composed of a criterion to minimize, and of a set of equality and inequality constraints that must be satisfied. Our chosen criterion is to minimize the regulation duration of the mission. The variables of our problem are for each task: (i) the time of its entry, (ii) the time of its removal (from the SoT), and (iii) the gains $(\lambda^I, \lambda^F, \beta)$ which describe the task execution behavior.

[2]contrary to Allen Logic, that only considers the start and end points of the time interval, here is also considered the regulation time t^R

The general optimization problem is written as follows:

$$\min_{\mathbf{x}} t^{\text{End}} \tag{6a}$$
$$\text{subject to } \dot{\mathbf{q}} = SoT_{\mathbf{x}}(\mathbf{q}, t) \tag{6b}$$
$$\text{seq}(\mathbf{q}) < 0 \tag{6c}$$
$$\phi(\mathbf{q}) < 0 \tag{6d}$$
$$\forall i, t_i^F \leq t^{\text{End}} \tag{6e}$$

The vector $\mathbf{x} = [t_1^I, t_1^F, \lambda_1^I, \lambda_1^F, \beta_1, \ldots, t_n^I, t_n^F, \lambda_n^I, \lambda_n^F, \beta_n, t^{\text{End}}]$ gathers the optimization variables of each task and t^{End}, the duration of the mission. $\text{seq}(\mathbf{q})$ and $\phi(\mathbf{q})$ are respectively the sequencing and the robotic constraints.

The optimization criterion t^{End} is computed indirectly. An equivalent explicit definition could be given by $t^{\text{End}} = \max_i(t_i^F)$. However this constraint is not smooth. Giving only (6b), the problem is smooth and properly defined: at the optimal solution, t^{End} will be equal to the maximum termination time of all tasks' t_i^F.

Vector $\mathbf{q}$ is in fact a vector of functions of time, hence constraints $\phi(\mathbf{q})$ are semi-infinite, *i.e.* taking place for all the values of the continuous variable $t \in [t^0, t^{\text{End}}]$.

It can be shown that (6) defines a continuous optimization problem. However, it cannot be solved directly because of the semi-infinite nature of the constraints. Therefore we expanded the semi-infinite constraint into a discreet form.

B. Constraints

Parameter $\mathbf{x}$ must satisfy both the sequencing and the robotic time-constraints enumerated hereafter:

1) Tasks constraints, noted seq$(\mathbf{q})$*:* gather the task sequence conditions of (5) and the following constraints:
For each task i:

Time coherence	$0 \leq t_i^I < t_i^F \leq t^{\text{End}}$	(7a)
Termination condition	$\|s_i^* - s_i(t_i^F)\| < \epsilon_i$	(7b)
Gain consistency	$\lambda_i^I \leq \lambda_i^F$	(7c)

The constraints (5a), (5c), (5e), (7a) and (7c) are linear. On the contrary, the constraint (7b) is impossible to compute directly using $\mathbf{x}$, and is determined from a *simulation* of the execution. Care has to be taken while resolving the condition described by (5b) and (5d). Indeed, discretizing t^R to the closest simulation step will produce discontinuities which may disturb the optimization process. A rather fastidious solution to this continuity problem would be to determine this point by interpolation. Another solution is to reformulate them by evaluating the regulation of the task i instead. The constraint (5b) and (5d) becomes respectively:

$$\forall (i,j) \in S_{R,I}, \|s_i^* - s_i(t_j^I)\| \leq \epsilon_i \tag{8a}$$
$$\forall (i,j) \in S_{R,F}, \|s_i^* - s_i(t_j^F)\| \leq \epsilon_i \tag{8b}$$

2) Robot constraints : $\phi(\mathbf{q})$: Those constraints are mainly due to hardware intrinsic limitations of the robot:

$$\text{Joint limits} \quad \mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max} \tag{9a}$$

$$\text{Velocity limits} \quad \dot{\mathbf{q}}_{\min} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_{\max} \tag{9b}$$

$$\text{Collision avoidance} \quad 0 \leq d_{ij} \tag{9c}$$

$\mathbf{q}_{\min}$, $\mathbf{q}_{\max}$, $\dot{\mathbf{q}}_{\min}$, $\dot{\mathbf{q}}_{\max}$ are respectively the lower and upper joint limits and the lower and upper velocity limits. d_{ij} is the distance between objects i and j. Object designate those found in the mission's environments and each link of the robotic system. Hence, both collision with the environment and self-collision of the robot have to be evaluated.

All of those constraints are semi-infinite: the following section presents how they have been tackled.

C. Technical aspects of the optimization resolution

1) Semi-infinite constraints: In a first approach, we tried to discretize the semi-infinite constraints on the basis of the simulation steps grid. However, since the number of the grid sample points changes in function of t^{End}, the number of constraints is variable. Subsequently a classical optimization solver can not handle them.

Let c be the evaluation value of a given constraint: $(\forall t \in [t^0, t^{\text{End}}], c(t) < 0)$. We considered associating only one value to the constraint, c_V, that is computed as follows: If the constraint is always satisfied, then c_V is the higher value of $c(t)$. Otherwise, it is the sum of all the violations found at each time step. Considering that the time step can change (e.g. when adding an interpolation point), we choose to weight the added value by the time step δt.

2) Constraint by task: Each task appears only once in the sequence, but a same action can be associated to many tasks. Associating the constraints $\phi(\mathbf{q})$ to the whole simulation can thus raise an issue: a violated constraint can not be linked to the responsible task. In order to compensate this problem, we consider n_T additional sets of constraint $\phi(\mathbf{q})$, noted $\phi_i(\mathbf{q}), i \in [1 \dots n_T]$, (with n_T the number of tasks in the sequence). Each set $\phi_i(\mathbf{q})$ is computed only when the task i is in the SoT.

3) Scaling: Since the constraints are not homogeneous (times, angles, velocities, distances), they have to be normalized based on the constraint values obtained while executing the sequence corresponding to the initial set of parameters $\mathbf{x}_0$. This simple scaling improves significantly the convergence of the optimization.

D. Absolute versus relative timing

In this parameterization, the tasks are described with an absolute time. As it is, decreasing t_i^I for a task i will not have any direct effect on t_i^F: we have also to decrease t_i^F then decrease t^{End}: it is thus necessary to propagate the reduction for all the following tasks. To avoid this, another parameterization consists in describing the SoT entry time of a given task with respect (i.e. relatively) to the previous one. We introduce a relative timing: each task is now described by two delays (instead of the absolute times t^I and t^F), namely:

1) dt^I: is the delay which occurs between (i) the maximum time of entry or of end of the preceding tasks, and (ii) the SoT entry time of the task in question.

$$t_i^I = \max\left(\max_{(j,i)\in S_{I,I}}\{t_j^I\}, \max_{(j,i)\in S_{F,I}}\{t_j^F\}\right) + dt^I \tag{10}$$

2) dt^F: is the delay between the SoT entry and the removal times of the task in question.

$$t_i^F = t_i^I + dt_i^F \tag{11}$$

Subsequently, the new parameter vector is noted : $\mathbf{x}' = [dt_1^I, dt_1^F, \lambda_1^I, \lambda_1^F, \beta_1, \dots, dt_n^I, dt_n^F, \lambda_n^I, \lambda_n^F, \beta_n, t^{\text{End}}]$.

If the task sequence is only a chain of tasks realized one after the other, we directly have $\mathbf{x}' = f(\mathbf{x})$, with f a linear function, and $t^{\text{End}} = \sum_i (dt_i^I + dt_i^F)$

Considering this new set of parameters, the formulation of the optimization problem changes: some tasks constraints of $\text{seq}(\mathbf{q})$ are modified. The previous constraints (5a), (5c) and (7a) are replaced by these constraints on the delay:

$$\forall i, 0 \leq dt_i^I \tag{12a}$$

$$\forall i, 0 < dt_i^F \tag{12b}$$

IV. Implementation

A. Optimization

At each optimization step, the solver chooses a new set of parameters $\mathbf{x}$. It then computes the constraints. Constraints (5e) and (7c), (12a) and (12b), can be evaluated directly. As stated previously, the other constraints can not be directly computed (since they do not write in an analytical formulation). They are thus evaluated using a complete simulation of their execution. The chosen value of the current optimization variable vector $\mathbf{x}$ is transmitted by the optimization solver to the simulation engine. The simulation returns the evaluation of the constraints and the optimization solver computes a new step vector $\mathbf{x}$, until convergence.

Our optimization problem is a non-linear contrained parametric problem. We chose the SQP algorithm from the MATLAB optimization toolbox, which is suitable to this kind of problem.

B. Simulation

In section II, we presented the computation of desired joint velocities for a hierarchy of tasks, as (6). The simulation is basically a numerical integration of this equation (we used an explicit Euler integration method with a fixed step $\Delta t = 0.005$sec). The entry and exit times t_i^I and t_i^F are continuous variables that are not aligned with the grid. Those instants are important since they correspond to a change in the SoT and thus a change in the control. If postponing the change of control to the next time step (like on a real system) we will not have a continuous problem (hence potentially raising the same problem described in section III-B). To solve this problem, the entry time t_a is

added as an integration point during the time step $[t, t+\Delta t]$, splitting it into the two smaller ones $[t, t_a]$ and $[t_a, t+\Delta t]$.

Initialization
$[t_1^I, t_1^F, \ldots, t_n^I, t_n^F] = \texttt{computeTimes}\,(\mathbf{x})$
$t_{\text{Sim}}^{\text{End}} = \max_i \left(t_i^F\right),\ t = 0$
while $(t < \max\left(t^{End}, t_{Sim}^{End}\right))$ **do**
  $\Delta t' = \texttt{findTimeStep(t)}$
  `handleStackOfTasks` (t)
  `updateConstraints()`
  $t = t + \Delta t'$
end

Algorithm 1: Tasks sequencing simulation

The algorithm 1 describes the simulation. The function `computeTimes` computes the absolute times using the relative times. The function `findTimeStep` computes the required time step for the Euler integration: the initial Δt, or a smaller one if needed, due to the need of splitting this interval in two. The function `handleStackOfTasks` computes the velocity of the robot induced by the tasks execution and integrates it, altogether with any other simulated objects or processes, to obtain the new positions.

The simulation engine runs under the AMELIF framework [4], an interactive dynamic simulator for virtual avatars which includes collision detection and task handling according to the SoT formalism. The execution for both simulation and real-robot control is performed by a generic control framework based on [12].

V. Experiment

A. Temporal network

The sequence of tasks (Fig. 2) describes a robot taking out a can from the fridge. The corresponding tasks are:

- Tasks of the right arm:
  $(\mathbf{e_0})$: open the gripper, $(\mathbf{e_1})$: move it to the fridge handle, $(\mathbf{e_2})$: close it, $(\mathbf{e_3})$: open the fridge, $(\mathbf{e_4})$: close the fridge
- Tasks of the left arm:
  $(\mathbf{e_5})$: open the gripper, $(\mathbf{e_6})$: move it in the fridge area, $(\mathbf{e_7})$: move it to the can, $(\mathbf{e_8})$: close it, $(\mathbf{e_9})$: lift the can, $(\mathbf{e_{10}})$: remove the can out of the fridge,

The task $\mathbf{e_6}$ is an intermediary task introduced as a way point: its tolerance on task regulation ϵ_6 is large so that the arm does not have to stop. This is part of the optimization decision, in order to reduce the execution time.

This is a complex mission that can not be split into smaller sequences. Indeed, the sequence is centered on the fridge: the grasping part does not make sense if the fridge is closed. Instead of adding explicit timing conditions between the tasks to ensure that this will never occur, we choose to consider as constraint the collision between the left arm and the door, in order to allow task overlapping.

The constraints considered for this problem are thus sequencing and robotic constraints (joint position and velocity limits), and collision avoidance with the fridge.

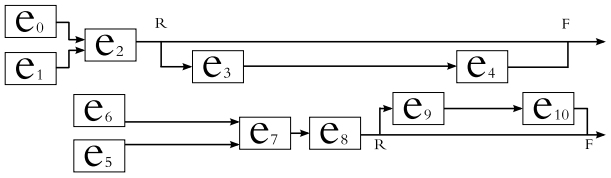

Fig. 2. Sequence describing the HRP-2 taking the can in the fridge

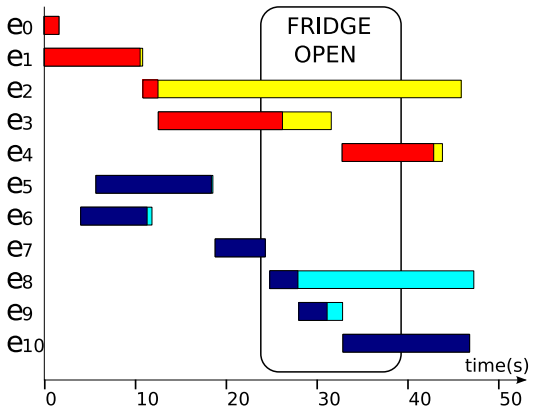

Fig. 3. Results of the optimization of the sequence of task: when the task is added in the SoT, its error is first regulated (this is the dark part (red or dark blue) of the block). From t_i^R, the error is nearly null and the task is kept in the SoT (light part (yellow or cyan) of the block) until t_i^F.

B. Results of the optimization

We ran the optimization on a 3GHz desktop PC running under Windows OS. The sequence found is described on Fig. 3. Each task is described by two periods: the dark one is the achievement period $[t_i^I, t_i^R]$, the bright one is the SoT presence period $[t_i^I, t_i^F]$.

The overlaps between the tasks of the left and the right arm appear clearly: the left arm starts to move before the fridge is open. It then starts to move toward the can pose even if the fridge is not completely open. And finally, the right arm starts to close the fridge before the left arm has completely left the fridge area. The whole task sequence lasts 47sec. Without these two overlaps, the robot will move to and grasp the can $(\mathbf{e_7})$ only after the fridge is fully opened $(\mathbf{e_3})$ and it will close the fridge $(\mathbf{e_4})$ only after the can is completely taken out $(\mathbf{e_{10}})$; consequently the total mission would have taken at least 71sec.

C. Experiment on the real robot

The task sequence is experimented on the upper body of the HRP-2 humanoid robot. Only the described tasks are used to compute the control law (which means that no additional care is taken for ensuring the constraints). For the tasks requiring a haptic interaction (i.e. opening and closing the fridge) the force sensor of the robot is used to close the loop and compensate for position uncertainties.

The robot manages to grasp the can without colliding any obstacle or joint limits, and respecting the given velocity limits. The obtained execution is plotted on Fig. 4. Thanks to the optimized gain, the convergence of the error of the tasks

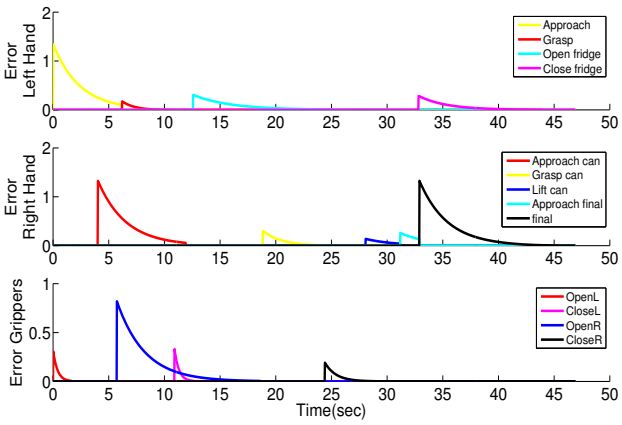

Fig. 4. Experiment on HRP-2: errors diminish when optimized task scheduling is applied: (top) right arm tasks (middle) left arm tasks (bottom) gripper tasks. The concurrency between the tasks is clearly visible.

that require a good precision (grasping the fridge handle and the can) is achieved as quickly as allowed by joint velocity limits. Snapshots of the execution are given in Fig. 5[3].

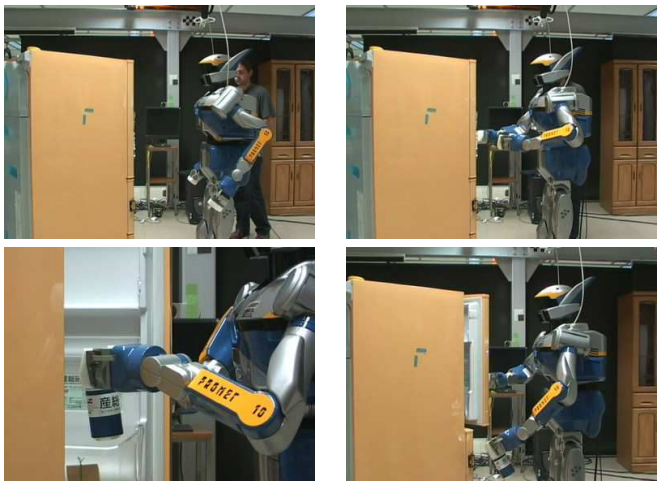

Fig. 5. HRP-2 grasping a can in the fridge.

VI. Conclusion

We devise a method which allows to optimize both the behavior and the overlapping scheduling of a sequence of tasks composing a robotic mission. The solution derives from an optimization formulation of the tasks scheduling keeping the formalism built on the top of a task-function based control. This allows to include the robot limitations as well as collision avoidance as constraints. Our method is exemplified through a complete simulation of a complex mission, where we demonstrated an improvement in the smoothness of the generated motion. For the time being, our method still needs a predefined ordered sequence. As a future work we will increase the autonomy by determining automatically the ordered sequence and compute all the necessary subtasks from definitions of actions/objects associations. We will also focus on more complex scenario using in particular perception tasks such as visual interaction.

[3] `www.laas.fr/~fkeith/iros09.avi`, the video is also attached to the paper

Acknowledgment

This work is partly supported by the European project FP6 ROBOT@CWE `www.robot-at-cwe.eu`.

References

[1] P. Baerlocher and R. Boulic. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20):402–417, Aug. 2004.
[2] R. Dechter. *Constraint Processing*, chapter 12, Temporal Constraint Network. Morgan Kaufmann, 2003.
[3] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. Robot. Autom.*, 8(3):313–326, 1992.
[4] P. Evrard, F. Keith, J.-R. Chardonnet, and A. Kheddar. Framework for haptic interaction with virtual avatars. In *IEEE Int. Symp. on Robot and Human Interact. Comm. (RO-MAN'08)*.
[5] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kauffmann Publishers, 2004.
[6] H. Hanafusa, T. Yoshikawa, and Y. Nakamura. Analysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress*, volume 4, pages 1927–1932, Kyoto, Japan, 1981.
[7] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1):43–53, Feb. 1987.
[8] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Trans. Robot.*, 2004.
[9] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
[10] S-H. Lee, J. Kim, F.C. Park, M. Kim, and J. Bobrow. Newton-type algorithms for dynamics-based robot movement optimization. *IEEE Transactions on Robotics*, 21(4):657–667, August 2005.
[11] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12):868–871, December 1977.
[12] N. Mansard and F. Chaumette. Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1):60–72, Feb. 2007.
[13] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi. Visually-guided grasping while walking on a humanoid robot. In *IEEE Int. Conf. Robot. Autom. (ICRA'07)*, pages 3041–3047, Roma, Italia, Apr. 2007.
[14] S. Miossec, K. Yokoi, and A. Kheddar. Development of a software for motion optimization of robots– application to the kick motion of the HRP-2 robot. In *ROBIO'06*, 2006.
[15] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *International Journal of Robotics Research*, 6(2):3–15, Feb. 1987.
[16] F. Py and F. Ingrand. Dependable execution control for autonomous robots. In *IEEE/RSJ Int. Conf. Intelligent Rob. Sys. (IROS'04)*, pages 1136–1141, Sendai, Japan, Sep. 2004.
[17] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and robot control. In *IEEE Int. Conf. Robot. Autom. (ICRA'93)*, volume 2, pages 802–807, Atlanta, USA, May 1993.
[18] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.
[19] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE Int. Conf. Robot. Autom. (ICRA'06)*, 2006.
[20] N. Sian, K. Yokoi, S. Kajita, F. Kanehiro, and K Tanie. A switching command-based whole-body operation method for humanoid robots. *IEEE/ASME Transactions on Mechatronics*, 10(5):546–559, Oct. 2005.
[21] B. Siciliano and J-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *IEEE Int. Conf. on Adv. Rob. (ICAR'91)*, 1991.