Optimized Time-Warping Tasks Scheduling for Smooth Sequencing

François Keith, Nicolas Mansard, Sylvain Miossec, Abderrahmane Kheddar

Optimized Time-Warping Tasks Scheduling for Smooth Sequencing

François Keith [*,****] **Nicolas Mansard** [**] **Sylvain Miossec** [***]
Abderrahmane Kheddar [*,****]

[*] *CNRS-LIRMM, Montpellier, France*
(email: {keith,kheddar}@lirmm.fr)
[**] *CNRS-LAAS, Toulouse, France (e-mail: nmansard@laas.fr)*
[***] *PRISME-Univ. d'Orléans, Bourges, France*
(e-mail: sylvain.miossec@bourges.univ-orleans.fr)
[****] *CNRS-AIST JRL, UMI3218/CRT, Tsukuba, Japan*

Abstract: This paper presents an optimal formulation for sequencing a set of robotic tasks. Tasks description and execution are based on the task formalism. A naive solution would be to arrange the execution of the tasks sequentially (potential redundancy not exploited, not optimal), or to set manually the timing and the behavior for each task (burdensome for the user, lack of autonomy, not optimal), or to arrange the tasks by priority and pill them up at once in the stack-of-tasks (high likelihood of conflicts, wrong or cumbersome execution). Our contribution is to determine the time-optimal realization of the mission taking into account robotic constraints as complex as collision avoidance; its originality lies in keeping the task formalism in the formulated optimization of the task sequencing problem. This theory is exemplified through a simulation on a 2D robot.

Keywords: Scheduling algorithms, Task-based control, Optimization, Robotics

1. 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. A typical example of such a mission decomposition is given in Fig. 1.

Numerous works have been proposed to obtain such a sequence of tasks from a given mission and a set of causal paradigms (Dechter (2003); Ghallab et al. (2004)). 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 (LaValle (2006)), that will compute the trajectories to be followed by the robot (Lamare and Ghallab (1998)). Yet, the meaning of the symbolic plan

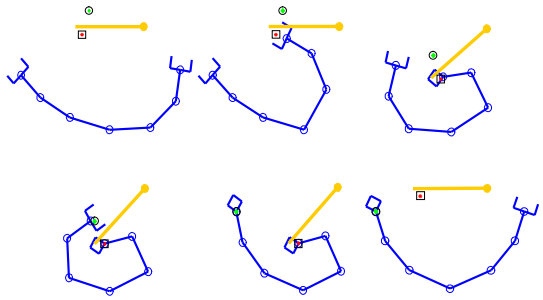

Fig. 1. Typical schedule plan for the execution of a given mission: in order to grasp the green object placed behind the planar door (the yellow line segment), the mission can be divided in six steps: (i) reach and grab the door's handle, (ii) open the door, (iii) reach the object, (iv) grasp it, (v) lift the object out and (vi) close the door. For this example, one link of the stick-robot is not allowed to move.

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 (Py and Ingrand (2004)) or distort locally the trajectory to apprehend small environment changes (Quinlan and Khatib (1993); Lamiraux et al. (2004)).

Rather than using a trajectory planner between the temporal reasoning and its robotic real execution, we suggest

⋆ This work is partially supported by grants from the ImmerSence EU CEC project, Contract No. 27141 `www.immersence.info` (FET-Presence) under FP6.

using a sensory-motor control approach based on task components. The task function (Samson et al. (1991)) or the operational space formulation (Khatib (1987)) 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 (Espiau et al. (1992); Chaumette and Hutchinson (2006)). 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) (Liégeois (1977); Hanafusa et al. (1981)). A secondary task can then be applied in the null-space, and, recursively, a hierarchic set of tasks (or *stack-of-tasks*) can be considered (Nakamura et al. (1987); Siciliano and Slotine (1991)). Hierarchy of tasks are becoming popular to build complex behavior for very redundant robot such as humanoids (Baerlocher and Boulic (2004); Bolder et al. (2007); Mansard et al. (2007); Sian et al. (2005); Sentis and Khatib (2006)). The formalism introduced in (Mansard and Chaumette (2007)) proved to be efficient in dealing with complex humanoid missions: the Stack of Tasks (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 2).

A task (*i.e.* a *task function* (Samson et al. (1991))) 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 saccadic 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 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 discreet form –i.e. the resources (mainly redundancy in motion generation) are not discreet–, task scheduling techniques can not apply directly to our problem. On the contrary, semi-infinite optimization techniques (Lee et al. (2005), Miossec et al. (2006)) can be used to generate low level trajectories for the overall execution, providing the problem to be formulated this way. One of the drawbacks of such an approach is that the generated trajectory does not necessarily account for the robot controller and is not robust to environment uncertainties: we propose a method that encapsulates the tasks concept to solve this issue.

The overall architecture, including the *optimized task sequencing*, is illustrated in Fig. 2. 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 2. The generic solution is then detailed in Section 3. The theory is finally exemplified through a simulation on a 2D robot, the mission consisting in getting an object placed behind a planar door.

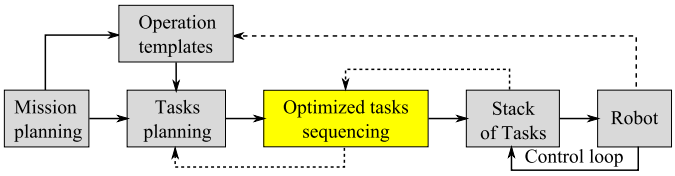

Fig. 2. Schematic representation of components of a robotic mission execution.

2. GENERIC TASK SEQUENCING

2.1 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 (Ben-Israel and Greville (2003)) 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 $\mathbf{P}$. A typical requested behavior is the regulation of the error, which can be obtained though 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 (Siciliano and Slotine (1991); Baerlocher and Boulic (2004); Sentis and Khatib (2006)): $\mathbf{z}$ 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:

$$\dot{\mathbf{q}}_\mathbf{i} = \dot{\mathbf{q}}_{\mathbf{i-1}} + (\mathbf{J_i}\mathbf{P}^\mathbf{A}_{\mathbf{i-1}})^+(\dot{\mathbf{e}}_\mathbf{i} - \mathbf{J_i}\dot{\mathbf{q}}_{\mathbf{i-1}}), \quad i = 1 \dots n \tag{4}$$

where $\dot{\mathbf{q}}_0 = \mathbf{0}$, $\mathbf{P}^\mathbf{A}_\mathbf{i}$ is the projector onto the null-space of the augmented Jacobian $\mathbf{J}^\mathbf{A}_\mathbf{i} = (\mathbf{J_1}, \dots, \mathbf{J_i})$ and $\widetilde{\mathbf{J}}_\mathbf{i} =$

$\mathbf{J_i P^A_{i-1}}$ is the limited Jacobian of the task i. The robot articular velocity realizing all the tasks in the stack is $\dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{n}$. A similar recursive formulation can be obtained to compute the $\mathbf{P^A_i}$, Baerlocher and Boulic (2004). The complete implementation of this approach, denoted SoT, is proposed in Mansard et al. (2007). The proposed 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.

2.2 Gain handling

The simple attractor presented in (3) produces a nice exponential decrease. However, it also produces a strong acceleration at the beginning of the task, 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 $\lambda = \lambda(\mathbf{e}(t))$, for example:

$$\lambda(\mathbf{e}) = (\lambda^F - \lambda^I) \exp\left(-\|\mathbf{e}\|\beta\right) + \lambda^I \tag{5}$$

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

2.3 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, in order 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. Thus we define the tolerance on the task regulation ϵ_i: a task is considered as regulated when $\|\mathbf{e_i}(t)\| \leq \epsilon_i$. The regulation time t_i^R is defined by $\|\mathbf{e_i}(t_i^R)\| = \epsilon_i$.

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 [1] : $\mathbf{e_i}$ must begin or end once $\mathbf{e_j}$ has begun, has ended or has been regulated. A graphical representation of such time-constraints is given in Fig. 3.

For example, first we have to grasp an object and maintain the force closure on it ($\mathbf{e_A}$) before we can move 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 ($\mathbf{e_A}$).

[1] 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

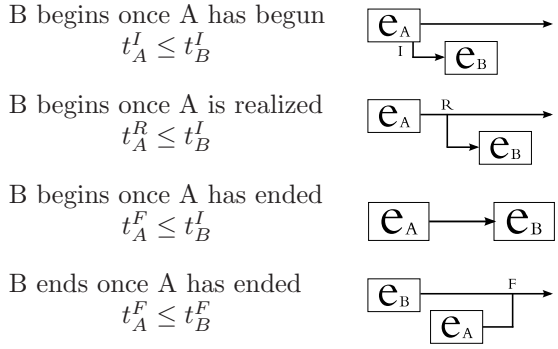

Fig. 3. Four time-dependency relations are considered.

We use the following notation to describe the set of pairs of tasks $\mathbf{e_i}$ and $\mathbf{e_j}$ that undergo these dependencies ($\mathbf{e_i}$ is the direct predecessor of $\mathbf{e_j}$) :

$$S_{I,I} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^I \leq t_j^I\} \tag{6a}$$
$$S_{R,I} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^R \leq t_j^I\} \tag{6b}$$
$$S_{F,I} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^F \leq t_j^I\} \tag{6c}$$
$$S_{F,F} = \{(\mathbf{e_i}, \mathbf{e_j}) \mid t_i^F \leq t_j^F\} \tag{6d}$$

3. CONTINUOUS OPTIMIZATION OF SEQUENCE OF TASKS

3.1 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 three for each task: (i) the time of its entry and (ii) of its removal (from the SoT), and (iii) the gains $(\lambda^I, \lambda^F, \beta)$, which describe the task execution behavior.

The general optimization problem is written as follows:

$$\min_{\mathbf{x}} t^{\text{End}} \tag{7a}$$
$$\text{subject to } \dot{\mathbf{q}} = SoT_{\mathbf{x}}(\mathbf{q}, t) \tag{7b}$$
$$\text{seq}(\mathbf{q}) < 0 \tag{7c}$$
$$\phi(\mathbf{q}) < 0 \tag{7d}$$
$$\forall i, t_i^F \leq t^{\text{End}} \tag{7e}$$

The vector $\mathbf{x}$ gathers the optimization variables of each task: $\mathbf{x} = [t_1^I, t_1^F, \lambda_1^I, \lambda_1^F, \beta_1, \ldots, t_n^I, t_n^F, \lambda_n^I, \lambda_n^F, \beta_n]$. t^{End} is 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 (7b), 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 $t \in [0, t^{\text{End}}]$.

It can be shown that (7) 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.

3.2 Constraints

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

Tasks constraints: noted seq($\mathbf{q}$) and defined as follow:

- Time coherence for each task i, that is:
$$\forall i, 0 \leq t_i^I < t_i^F \leq t^{\text{End}} \tag{8}$$
- Termination condition for each task i, that is:
$$\forall i, \|s_i^* - s_i(t_i^F)\| < \epsilon_i \tag{9}$$
- Any task sequence condition for a given couple of tasks i and j described in (6)
- The control gain, namely:
$$\forall i, \lambda_i^I \leq \lambda_i^F \tag{10}$$

The constraints (6a), (6c), (6d) and (8) are linear. On the contrary, the constraint (9) 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 (6b). 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 (6b) and evaluate the regulation of the task i when the task j begins. The constraint (6b) becomes:

$$\forall (i,j) \in S_{R,I}, \|s_i^* - s_i(t_j^I)\| \leq \epsilon_i \tag{11}$$

Robot constraints: $\phi(\mathbf{q})$

Those constraints are mainly due to hardware intrinsic limitations of the robot:

- Joint limits, given by
$$\mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max} \tag{12}$$
$\mathbf{q}_{\min}$, $\mathbf{q}_{\max}$ are respectively the lower and upper joint limits.
- Velocity limits, given by
$$\dot{\mathbf{q}}_{\min} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_{\max} \tag{13}$$
$\dot{\mathbf{q}}_{\min}$, $\dot{\mathbf{q}}_{\max}$ are respectively the minimal and maximal velocity limits for each joint.
- Collision avoidance between a given pair of objects i and j, given by
$$d_{ij} \geq 0 \tag{14}$$
d_{ij} is the distance between objects i and j. Objects 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 Benallegue et al. (2009).

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

3.3 Technical aspects of the optimization resolution

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 it.

Let c be the evaluation value of a given constraint: ($\forall t \in [t^I, t^{\text{End}}], c(t) < 0$). 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. The reformulation $c_V < 0$ acts similarly to the semi-infinite constraint $c(t)$.

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.

3.4 Absolute versus relative timing

In this parameterization, the tasks and constraints 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, of end or of regulation of the preceding tasks, and (ii) the SoT entry time of the task in question.
(2) dt^F: is the delay between the SoT entry and the removal times of the task in question.

These two delays fulfil the following equations:

$$t_j^I = \max\left(\max_{(i,j)\in S_{I,I}}\{t_i^I\}, \max_{(i,j)\in S_{R,I}}\{t_i^R\}, \max_{(i,j)\in S_{F,I}}\{t_i^F\}\right) + dt^I \tag{15}$$

$$t_i^F = t_i^I + dt_i^F \tag{16}$$

Subsequently, the new parameter vector is noted :
$\mathbf{x}' = [dt_1^I, dt_1^F, \lambda_1^I, \lambda_1^F, \beta_1, \ldots, dt_n^I, dt_n^F, \lambda_n^I, \lambda_n^F, \beta_n]$.

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 $t^{\text{End}} = \sum_i dt_i^I + \sum_i dt_i^F$

Considering this new set of parameters, the formulation of the optimization problem changes. The objective remains the same: minimize t^{End}, while the constraints become:

- Constraints on the delay:
$$\forall i, 0 \leq dt_i^I \tag{17}$$
$$\forall i, 0 < dt_i^F \tag{18}$$
- Constraint of termination
$$\forall i, \|s_i^* - s_i(t_i^F)\| \leq \epsilon \tag{19}$$

- Constraint on the task sequence

$$\forall (i,j) \in S_{F,F}, t_i^F \leq t_j^F \tag{20}$$

The constraints (6b), (6c) and (8) are now replaced by the constraints (17) and (18). Each task's start depends on its predecessors in the sequence. Because of the possible dependence on the regulation time, the absolute times t_i^I, t_i^R and t_i^F can not be pre-computed, and are computed during the simulation.

4. IMPLEMENTATION

4.1 Presentation

At each optimization step, the solver chooses a new set of parameters $\mathbf{x}$. It then computes the constraints. Constraints (17) and (18) 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. The optimization solver is chosen to be the SQP algorithm from the MATLAB optimization toolbox; the simulation engine runs also under MATLAB.

4.2 Simulation

In Section 2, we presented the computation of the joint-velocities control law (7). The simulation is basically a numerical integration of this last equation (using an explicit Euler integration method with a fixed step $\Delta t = 0.005$sec). The starting t_i^I and ending t_i^F times of tasks 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 3.2). To solve this problem, the entry time t_a for a given task is added as an integration point during the time step $[t, t+\Delta t]$, resulting in the separation of the time step into the two smaller time steps $[t, t_a]$ and $[t_a, t+\Delta t]$.

Initialization
$t^{\text{End}} = t^{\text{End}}_{\max} = \sum_i dt_i^I + \sum_i dt_i^F$
$t = 0$
while $(t < t^{End})$ **do**
  $[t_1^I, t_1^R, t_1^F, \ldots, t_n^I, t_n^R, t_n^F, \ldots, t^{\text{End}}] =$ `updateTimes`(t)
  $\Delta t' =$ `findTimeStep`(t)
  `handleStackOfTasks` (t)
  `updateConstraints`()
  $t = t + \Delta t'$
end
return t^{End}

Algorithm 1: Tasks sequencing simulation

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 to the tasks execution and integrates it, altogether with any other simulated objects or processes, to obtain the new positions.

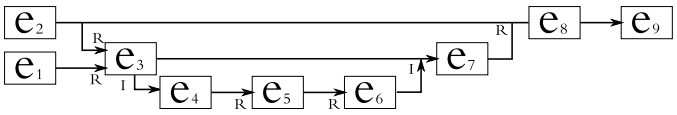

Fig. 4. Sequence describing the robot taking the object

5. EXPERIMENT

5.1 Temporal network

The sequence of tasks (Fig. 4) describes a robot taking out an object placed behind a door. The corresponding tasks are:

- $\mathbf{e_1}$ Move the right arm to the door
- $\mathbf{e_2}$ Close the right gripper
- $\mathbf{e_3}$ Open the door
- $\mathbf{e_4}$ Move the left gripper to the object
- $\mathbf{e_5}$ Close the left gripper
- $\mathbf{e_6}$ Remove the object out
- $\mathbf{e_7}$ Close the door
- $\mathbf{e_8}$ Open the right gripper
- $\mathbf{e_9}$ Move the right arm away

This is a complex mission that can not be split into smaller sequences. Indeed, the sequence is centered on the door: the grasping part does not make sense if it is closed. Instead of adding an explicit timing conditions between the tasks to ensure that this will never occur, we chose to consider as constraint the collision between the left arm and the door, in order to allow task overlapping (the condition $t_3^I \leq t_4^I$ only postpones the entry time of the task without preventing overlapping). Similarly, we considered the collision constraint between the gripper and the handle or the object. The gripper closing tasks are used to grasp and hold objects, so these tasks have to be maintained as long as necessary.

The constraints considered for this problem are thus sequencing and robotic constraints (joint position and velocity limits).

5.2 Results of the optimization

We ran the optimization on a 3GHz desktop PC running under Windows OS. No specific effort of software optimization has been made. The sequence found is described on Fig. (5).

The overlaps between the tasks of the left and the right arm appear clearly: the left arm starts to move before the door is open. It then starts to move toward the object pose even if the door is not completely open. And finally, the right arm starts to close the door before the left arm has completely left the door area. The whole task sequence lasts 46.9sec. Without these two overlaps, the robot will (i) move to and grasp the object ($\mathbf{e_4}$) only after the door is fully opened ($\mathbf{e_3}$) and it will close the door ($\mathbf{e_7}$) only after the object is completely taken out ($\mathbf{e_6}$); then the total mission would have taken 54sec.

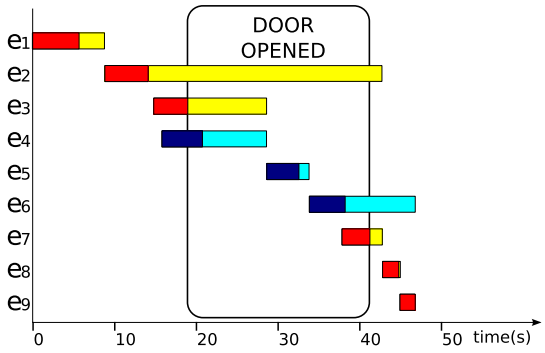

Fig. 5. Result of tasks sequencing and behavior optimization. Each task is described by two periods: the dark one is the achievement period $[t_i^I, t_i^R]$, the bright one is the SoT presence period $[t_i^I, t_i^F]$.

The obtained execution is plotted on Fig. 6 which underlines the relation between the norm of error of the tasks and velocity of the robot joints.

6. 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, doable by a humanoid robot, using in particular visual interaction.

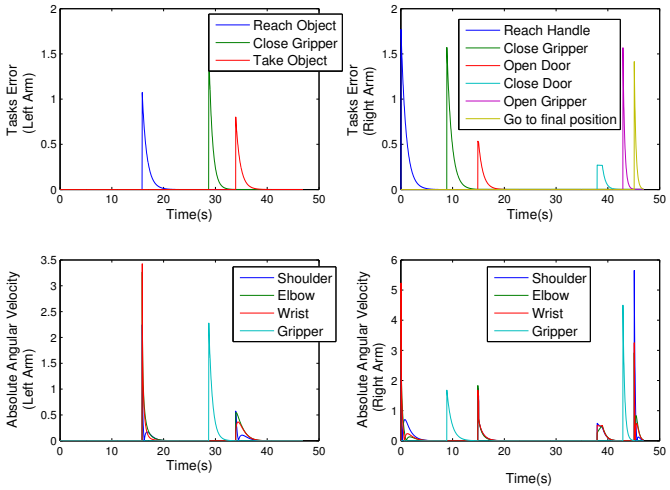

Fig. 6. Simulation: decrease of the errors when the final sequence is run

REFERENCES

Baerlocher, P. and Boulic, R. (2004). An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20), 402–417.

Ben-Israel, A. and Greville, T. (2003). *Generalized inverses: theory and applications*. CMS Books in Mathematics. Springer, 2nd edition.

Benallegue, M., Escande, A., Miossec, S., and Kheddar, A. (2009). Fast C^1 proximity queries using support mapping of sphere-torus-patches bounding volumes. In *IEEE International Conference on Robotics and Automation*, 483–488. Kobe, Japan.

Bolder, B., Dunn, M., Gienger, M., Janssen, H., Sugiura, H., and Goerick, C. (2007). Visually guided whole body interaction. In *IEEE Int. Conf. Robot. Autom. (ICRA'07)*, 3054–3061. Roma, Italia.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control, pt i: Basic approaches. *IEEE Robot. and Autom. Mag.*, 13(4), 82–90.

Dechter, R. (2003). *Constraint Processing*, chapter 12, Temporal Constraint Network. Morgan Kaufmann.

Espiau, B., Chaumette, F., and Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3), 313–326.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Morgan Kauffmann Publishers.

Hanafusa, H., Yoshikawa, T., and Nakamura, Y. (1981). Analysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress*, volume 4, 1927–1932. Kyoto, Japan.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1), 43–53.

Lamare, B. and Ghallab, M. (1998). Integrating a temporal planner with a path planner for a mobile robot. In *Proc. AIPS Workshop on Integrating planning, scheduling and execution in dynamic and uncertain environments*, 144 –151.

Lamiraux, F., Bonnafous, D., and Lefebvre, O. (2004). Reactive path deformation for nonholonomic mobile robots. *IEEE Trans. on Robotics*, 7(20), 967–977.

LaValle, S. (2006). *Planning Algorithms*. Cambridge Univ. Press.

Lee, S.H., Kim, J., Park, F.C., Kim, M., and Bobrow, J.E. (2005). Newton-type algorithms for dynamics-based robot movement optimization. *IEEE Transactions on Robotics*, 21(4), 657– 667.

Liégeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12), 868–871.

Mansard, N. and Chaumette, F. (2007). Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1), 60–72.

Mansard, N., Stasse, O., Chaumette, F., and Yokoi, K. (2007). Visually-guided grasping while walking on a humanoid robot. In *IEEE Int. Conf. Robot. Autom. (ICRA'07)*, 3041–3047. Roma, Italia.

Miossec, S., Yokoi, K., and Kheddar, A. (2006). Development of a software for motion optimization of robots– application to the kick motion of the HRP-2 robot. In *IEEE International Conference on Robotics and Biomimetics*.

Nakamura, Y., Hanafusa, H., and Yoshikawa, T. (1987). Task-priority based redundancy control of robot manipulators. *International Journal of Robotics Research*, 6(2), 3–15.

Py, F. and Ingrand, F. (2004). Dependable execution control for autonomous robots. In *IEEE/RSJ Int. Conf. Intelligent Rob. Sys. (IROS'04)*, 1136–1141. Sendai, Japan.

Quinlan, S. and Khatib, O. (1993). Elastic bands: Connecting path planning and robot control. In *IEEE Int. Conf. Robot. Autom. (ICRA'93)*, volume 2, 802–807. Atlanta, USA.

Samson, C., Le Borgne, M., and Espiau, B. (1991). *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, UK.

Sentis, L. and Khatib, O. (2006). A whole-body control framework for humanoids operating in human environments. In *IEEE Int. Conf. Robot. Autom. (ICRA'06)*, 2641–2648. Orlando, USA.

Sian, N., Yokoi, K., Kajita, S., Kanehiro, F., and Tanie, K. (2005). A switching command-based whole-body operation method for humanoid robots. *IEEE/ASME Trans. Mechatronics*, 10(5), 546–559.

Siciliano, B. and Slotine, J.J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *IEEE Int. Conf. on Advanced Robotics (ICAR'91)*, volume 2, 1211–1216. Pisa, Italy.