From Discrete Mission Schedule to Continuous Implicit Trajectory using Optimal Time Warping

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 generic solution to apply a mission described by a sequence of tasks on a robot while accounting for its physical constraints, without computing explicitly a reference trajectory. A naive solution to this problem would be to schedule the execution of the tasks sequentially, avoiding concurrency. This solution does not exploit fully the robot capabilities such as redundancy and have poor performance in terms of execution time or energy. 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.

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 into a set of tasks (i.e. generic sensory-motor functions) which has to be mapped into robot execution. Numerous works have been proposed to compute such a sequence of tasks from a given mission and a set of causal paradigms (Dechter 2003; Ghallab, Nau, and Traverso 2004; Li and Williams 2008). However, they generally produce a symbolic plan, where the only numerical precisions lie on the scheduled time data. Moreover, constraints have to be expressed under a symbolic expression. 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 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 often specifically hard coded) to enhance the trajectory with symbolic data, that would help re-computing only part of the plan (Py and Ingrand 2004) or distort locally the trajectory after small environment changes (Quinlan and Khatib 1993).

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 (Samson, Le Borgne, and Espiau 1991) is an elegant approach to produce intuitively sensor-based robot objectives. Based on the redundancy of the system, the approach can be extended to consider a hierarchical set of tasks (Siciliano and Slotine 1991). Hierarchy of tasks are becoming popular to build complex behavior for very redundant robot such as humanoids (Mansard and Chaumette 2007; Sentis and Khatib 2006).

A task (*i.e.* a *task function*) can be directly linked to the symbols on which the task temporal network is reasoning. Mission decomposition is thus executable directly using the sensory-motor mapping of the task function. However, exclusive task sequencing on the robot produces generally jerky suboptimal movements which may look to humans as monotonous automated motions. This paper focuses on finding a solution to produce automatically an optimal plane/schedule that makes use of the redundancy by enabling task concurrency. It seems difficult to use temporal networks to produce a scheduling with task overlapping **when the tasks concurrency is restricted by physical limitations** of the robot (for example obstacles or balance of a biped robot), since the constraints are not in a discreet form. On the other hand, semi-infinite optimization (Miossec, Yokoi, and Kheddar 2006) is known to generate low level trajectories, while accounting for such constraints, but with insufficient robustness to environment uncertainties.

In this paper, we propose to rely on task for both the symbolic reasoning and control of the robot. In between, we propose to use semi-infinite optimization to refine the symbolic schedule and account for system constraints. Given a sequence of tasks to achieve a mission, our solution returns for each task the optimal times at which it is activated and inactivated and the optimal parameters for the task execution. The originality of our approach lies in keeping the task component in the 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.

Generic Task Sequencing

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 $\dot{\mathbf{q}}$: $\dot{\mathbf{e}} = \mathbf{J}\dot{\mathbf{q}}$. Considering a reference behavior $\dot{\mathbf{e}}^*$ to be executed in the task space, typically,

$$\dot{\mathbf{e}}^* = -\lambda \mathbf{e}, \tag{1}$$

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. $\mathbf{P}$ ensures a decoupling of the task with respect to $\mathbf{z}$, which can be extended recursively to a set of n tasks, each new task being fulfilled without disturbing the previous ones:

$$\dot{\mathbf{q}}_\mathbf{i} = \dot{\mathbf{q}}_{\mathbf{i-1}} + (\mathbf{J_i}\mathbf{P}^\mathbf{A}_{\mathbf{i-1}})^+(\dot{\mathbf{e}}_\mathbf{i} - \mathbf{J_i}\dot{\mathbf{q}}_{\mathbf{i-1}}), \;\; i = 1 \dots n \tag{3}$$

where $\dot{\mathbf{q}}_0 = \mathbf{0}$ and $\mathbf{P}^\mathbf{A}_\mathbf{i}$ is the projector onto the null-space of the augmented Jacobian $\mathbf{J}^\mathbf{A}_\mathbf{i} = (\mathbf{J_1}, \dots \mathbf{J_i})$. The robot joint velocity realizing all the tasks in the stack is $\dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{n}$. A complete implementation of this approach is proposed in (Mansard and Chaumette 2007) 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 locally taken into account. The continuity of the control law is preserved even at the instant of change.

Gain handling

The simple attractor presented in (1) 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:

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

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

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.

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 $[t_i^I, t_i^F]$, defined with respect to t^0. The regulation time t_i^R is the time line after which the task error is considered as "regulated", i.e. sufficiently close to 0. It is defined by $\|\mathbf{e_i}(t_i^R)\| = \epsilon_i$, with ϵ_i the tolerance upon task realization.

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. We use 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 $\mathbf{e_j}$) :

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

Continuous optimization of sequence of tasks

Given a set of hypothesis described using (5), a generic solution is now proposed to automatically compute the optimal task and schedule parameters to be executed by the SoT.

General problem formulation

An optimization problem is composed of a criterion to minimize, along with a set of variables and equality and inequality constraints to be satisfied. Our criterion is the total duration of the mission. The variables are: (i) the times t_i^I and t_j^F and (ii) the gains $(\lambda^I, \lambda^F, \beta)$. The general optimization problem is written as follows:

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

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

Constraint definitions

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

[1] contrary to Allen Logic, that only considers the start and end date, here is also considered the regulation time t^R

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

- Time coherence for each task i, that is:
$$\forall i, 0 \leq t_i^I < t_i^F \leq t^{\text{End}} \tag{7}$$
- Termination condition for each task i, that is:
$$\forall i, \|s_i^* - s_i(t_i^F)\| < \epsilon_i \tag{8}$$
- Task sequence conditions desbribed in (5).
- Gain consistency for each task i, namely:
$$\forall i, \lambda_i^I \leq \lambda_i^F \tag{9}$$

The constraints (5a), (5d), (5e), (7) and (9) are linear. On the contrary, the constraint (8) 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 (5c). Indeed, discretizing t^R to the closest simulation step will produce discontinuities which may disturb the optimization process. The constraint (5b) and (5c) are thus equivalently rewritten:

$$\forall (i,j) \in S_{R,I}, \|s_i^* - s_i(t_j^I)\| \leq \epsilon_i \tag{10}$$
$$\forall (i,j) \in S_{R,F}, \|s_i^* - s_i(t_j^F)\| \leq \epsilon_i \tag{11}$$

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 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 the minimal and maximal velocity.
- Collision 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. Object designate those found in the mission environments and each link of the robotic system. Hence, both collision with the environment and self-collision of the robot have to be.

It can be shown that (6) defines a continuous optimization problem. However, $\mathbf{q}$ 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 $t \in [t^0, t^{\text{End}}]$. Cares have thus to be taken when implementing them in classical solver softwares for parameter optimization problems.

Implementation

Optimization

At each optimization step, the solver chooses a new set of parameters $\mathbf{x}$. It then computes the constraints. Constraints (5a), (5e), (7) and (9) can be evaluated directly. As stated previously, the other constraints can not be directly computed (since they do not write in an analytical explicit 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 solver to a simulation engine. The simulation returns the evaluation of the constraints and the optimization solver computes a new step vector $\mathbf{x}$, until convergence.

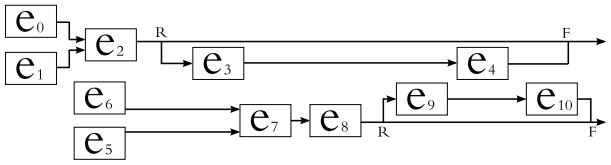

Figure 1: Sequence of tasks for taking the can in the fridge

Simulation

The simulation is basically a numerical integration of (6) using an explicit Euler integration method with a fixed step $\Delta t = 0.005$sec. The times t_i^I and t_i^F are continuous variables that may not be aligned with the integration grid, and have to be explicitly added to ensure continuity with respect to the optimization variables.

The simulation engine runs under the AMELIF framework (Evrard et al. 2008), 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 (Mansard et al. 2009).

Experiment

Temporal network

The experiments have been realized using a pick-in-box scenario: the robot has to open the fridge with one hand before grasping the object with the other hand. Similarly, it has to wait for the grasping to be completed before closing the fridge. For the particular manipulation detailled below, we can simply consider the collision detection to ensure the causality, instead of explicitly defining these causal constraints. The optimal schedule should use the system redundancy by overlapping the tasks on each arm.

The task sequence is composed of 10 tasks (see Fig.1): $\mathbf{e_0}$ - open the right gripper ; $\mathbf{e_1}$ - move the right arm to the fridge ; $\mathbf{e_2}$ - close the right gripper ; $\mathbf{e_3}$ - open the fridge ; $\mathbf{e_4}$ - close the fridge ; $\mathbf{e_5}$ - open the left gripper ; $\mathbf{e_6}$ - move the left gripper in the fridge area ; $\mathbf{e_7}$ - move the left gripper to the can ; $\mathbf{e_8}$ - close the left gripper ; $\mathbf{e_9}$ - lift the can ; $\mathbf{e_{10}}$ - remove the can out of the fridge.

This mission can not be split into smaller independent sequences. The constraints considered for this problem are thus sequencing and robotic constraints (joint position and velocity limits), and non-colliding with the fridge.

Results of the optimization

The optimization ran on a 3GHz desktop PC running under Windows OS, using MATLAB SQP solver. The sequence found is described on Fig. 2. The overlaps between the tasks of each arm appear clearly: the left arm starts moving before the fridge is open. It then starts the reaching motion even if the fridge is not completely open. 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 optimal timing will only be 71sec.

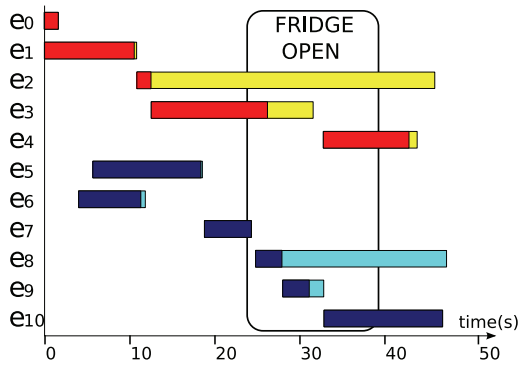

Figure 2: Results of the optimization of the sequence of task: when the task is added in the SoT, its error is first regulated (this corresponds to the dark part (red or dark blue) of the block). From t_i^R, the error is nearly null and the task is kept in the SoT (light part (yellow or cyan) of the block) until t_i^F.

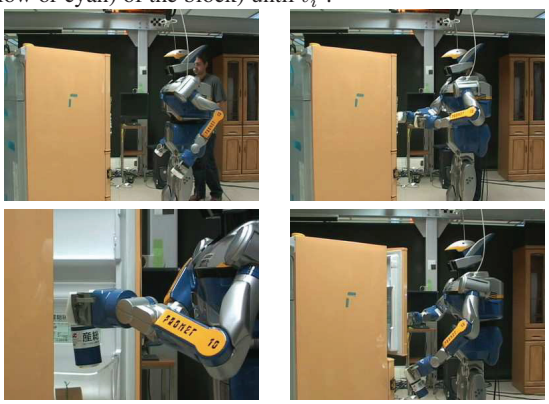

Figure 3: HRP-2 grasping a can in the fridge.

The SoT formalism allows to directly apply the optimized task sequence as a control law using the same task definition. The task sequence is executed by the robot HRP-2, a full-size humanoid robot with 30 actuated DOF. 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 constraint violation, in a smooth manner thanks to the optimized gain. Snapshots of the execution are given in Fig. 3. See `http://www.laas.fr/~nmansard/keithfridge.avi` for the video of the complete execution.

The complexity of the optimization does not depend on the number of possible solutions for a set of tasks, but on the number of parameters. Our solver has a linear complexity in the number of parameters $\mathcal{O}(\mathcal{P})$. For the demonstrated tasks (66 parameters and 120 constraints), one simulation takes around 1secs. The whole optimization process required 7 hours, mainly due to a first simplistic optimization scheme (in particular, gradients are estimated by finite differences). Reducing this cost is one of our main concerns.

Conclusion

We devise a method which allows to automatically pass from a symbolic plan to a complete motion generator that takes into account all kind of robot constraints, such as joint limits or collision. The result 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. The solution is generic, and may be applied to each system with time-modulable tasks allowing blending with continuous semi-infinite constraints. For the time being, our method still needs a predefined ordered sequence. As a future work the autonomy will be improved by determining automatically the ordered sequence and compute all the necessary subtasks from definitions of actions/objects associations.

Acknowledgment

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

References

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

Evrard, P.; Keith, F.; Chardonnet, J.-R.; and Kheddar, A. 2008. Framework for haptic interaction with virtual avatars. In *Robot and Human Interactive Communication (RO-MAN'08)*.

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

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.

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

Li, H. X., and Williams, B. C. 2008. Generative planning for hybrid systems based on flow tubes. In *ICAPS*, 206–213.

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

Mansard, N.; Stasse, O.; Evrard, P.; and Kheddar, A. 2009. A versatile generalized inverted kinematics implementation for collaborative humanoid robots: The stack of tasks. In *Submitted to International Conference on Advanced Robotics (ICAR'09)*.

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*.

Py, F., and Ingrand, F. 2004. Dependable exec. control for auton. robots. In *IEEE/RSJ Int. Conf. Intelligent Rob. Sys. (IROS'04)*.

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.

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.

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)*.