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

Contact Planning for Acyclic Motion with Tasks Constraints

Adrien Escande and Abderrahmane Kheddar

Abstract— **This paper extends our previous work on contact points planning in two ways. First, by taking advantage of the possibilities offered by our initial posture generator, we include additional tasks that are not related to locomotion within the planning. The output motion will be generated so as to cope with these tasks. Second, we refine the potential function that guides the planner by introducing potential fields acting each on a single body. This helps escaping local minima of the original potential field and thus to deal with more challenging scenarios. We then test these novelties on difficult problems with success, and experiment the output of one of the planned scenario on a HRP-2 humanoid robot.**

I. Introduction

Humanoids are anthropomorphic advanced robotic systems that are designed to mimic human functions in either collaborative (with humans) or standalone modes. Their highly redundant structure allows, in principle, to design rich motion behaviors and precise tasks in environments and application working contexts where they can apply. Despite this redundancy and the advanced hardware achievements, they are still in the infancy stage of their full exploitation and capabilities. This is because the problems of integration, fast planning, active perception and reactive behaviors are (among others) generally well understood but yet highly complex and unsolved. One of the prime functionality of humanoids is their ability to move themselves from a place to another. To tackle this problem, very robust cyclic walking patterns, even with reactive capabilities, exist. The ASIMO humanoid robot, and recently Toyota's partner humanoid can even run. Walking on uneven terrain has also been splendidly demonstrated for the quadrupedal BigDog robot; such demonstration can not be achieved yet on a humanoid.

Cyclic motions, although useful in practice, prove to be limited and there are plenty of situations where acyclic motion with eventually additional links' support are required to achieve complex transportation motions. A simple example could be the humanoid robot grasping a ramp to ease climbing high stairs. This has been demonstrated in [1]. Non-gaited motion planning has been more formally addressed in simulation in [2] and [3]. A different approach has been proposed by the authors in [4] and [5] where real experiments have been conducted on the HRP-2 humanoid robot. Acyclic motions having contact supports that could occur on any part of the robot with any part of the environment is on the stage of full realization, but there remain many problems to be solved and future extension to be addressed that have been discussed in [6].

Here, we show that our approach allows also an interesting extension at nearly no cost to handle additional task constraints that are not explicitly related to locomotion. For instance, assume that we want not only to plan the sequence of supporting contact points between which trajectories are generated to go from a given location to another, but also to hold a container full of liquid –such as a cup of Japaneses green tea– and keep it with a constant orientation all along the motion. We present a method to realize such scenario in nearly straightforward way since our planner relies on an optimization-based posture generator that can be seen as a generalized inverse kinematics under constraints. Once the contact sequences are planned, ideally, the trajectories are generated from an optimization software (e.g. [7]) and can be played with a stack-of-tasks reactive controller such as the one proposed in [8] or in [9].

We also present an improvement of the way we build the potential field that guides the planner. This makes the robot more likely to take larger steps when possible, resulting in a faster planner. It also allows to deal efficiently with local minima that inevitably come with important changes in the overall posture (e.g. when transiting from a two-legged locomotion to a four legged one). Such changes are required in bulky environments.

Firstly, we briefly recall the background of our contact planner, however we strongly recommend the reader to refer to our previous published work in [4] and [5]. This is followed by the way additional tasks constraints can be added and dealt with during planning. We then describe the improvement in the way of building the potential field. After it, the proposed ideas are exemplified through several scenarios including one where the HRP-2 is asked to bring a glass from some location, go toward the table, sit on the chair at this table, while keeping the glass of water vertically. Finally, this experiment is realized by the HRP-2 robot.

II. Background

A. Overall planner

In [4] we presented a planner with the following principle: planning is made in the space of sets of contacts SC, by building incrementally a tree of such sets. The difference between a father and its son is exactly one contact (one more or one less). To drive the planning, a potential function f is given. At each iteration, the best leaf of the tree (according to f) is selected and its sons are built by adding or removing a contact. If some of the new leaves are too close to existing nodes or leaves, they are discarded. This mechanism

A. Escande is with CEA-LIST, Fontenay-aux-Roses, France adrien(dot)escande(at)cea(dot)fr
A. Kheddar is with CNRS-UM2 LIRMM, Montpellier, France and CNRS-AIST JRL, UMI3218/CRT, Tsukuba, Japan kheddar(at)ieee(dot)org

is inspired by the potential-field-based planner Best First Planning (BFP), see [10]. However, we are planning here in SC, which allows substantial reduction of the search space compared to the usual configuration space. Yet it does not allow to take into account the geometrical and physical limitations of the robot: two contacts of a set may be too far from each another, a contact may force collisions or instability of the robot, etc. Feasibility of a set must be checked, and this is done with a posture generator (section II-B), see Fig. 1. Upon failure of the posture generator, the set of contacts is discarded.

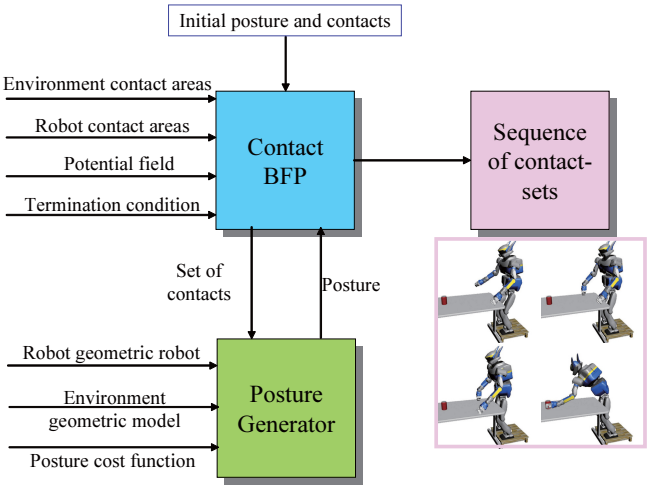

Fig. 1. Overview of the planner and its components.

In [5], we introduced some major concepts regarding the choice of new contacts, the design of the potential function, and a positive interaction between both of these concepts. The choice of a new contact is made directly during the posture generation attempt and is based on the criterion of the posture generator. We thus reduce significantly the number of feasibility checks to perform and especially the number of failures of generation attempts, which are the most time costly computations.

The design of f relies on a rough trajectory T in the configuration space. This trajectory is defined by several key postures between which a linear interpolation is made. It does not necessarily lies completely in the free configuration space, nor does it need to be in the robot stability space. It is just a guide upon which we build a descending valley-like potential whose minimum is at the end of T. We developed a method to generate this trajectory automatically, see [11].

The last point in [5] is to use f as part of the criteria in posture generation, and take this into account in the BFP-like part of the planner to generate far less nodes.

Planning is thus made in the sets of contacts space, but with a constant link to the configuration space. The inputs of our planner are the data of the environment, the data of the robot, a feasible starting set of contacts and some end conditions. Output is a sequence of sets of contacts along with their associated witness postures.

B. Posture Generator

For a given set of contacts $\{C_i\}$ as input, the posture generator writes and attempts to solve an optimization problem with (non-linear) constraints, whose variables are the n degrees of freedom of the robot $\mathbf{q}$:

$$\min_{\mathbf{q}\in\mathcal{Q}} f(\mathbf{q}) \tag{1}$$

where Q is the set of admissible postures for these contacts:

$$\mathcal{Q} = \begin{cases} q_i^- \leq q_i \leq q_i^+, \quad \forall i \in [|1,n|] & (2)\\ \epsilon_{ij} \leq d(r_i(\mathbf{q}), r_j(\mathbf{q})), \quad \forall (i,j) \in \mathcal{I}_{\text{auto}} & (3)\\ \epsilon_{ik} \leq d(r_i(\mathbf{q}), O_k), \quad \forall (i,k) \in \mathcal{I}_{\text{coll}} & (4)\\ s(\mathbf{q}) \leq 0 & (5)\\ g_i(\mathbf{q}) = 0 \quad \forall\ C_i & (6)\\ h_i(\mathbf{q}) \leq 0 \quad \forall\ C_i & (7) \end{cases}$$

Inequalities (2) are the joint limits; eqs. (3) define constraints for auto-collision avoidance between pairs of robot's bodies (r_i, r_j) given by $\mathcal{I}_{\text{auto}}$. d is the minimum distance between two objects, that need to be positive; eqs. (4) deal also with collision avoidance between r_i and any object O_k of the environment, pairs are defined by $\mathcal{I}_{\text{coll}}$; ϵ_{ij} and ϵ_{ik} are security margins for these constraints; s is a static stability criterion – s can simply be the belonging of the projection of the center of mass to the convex hull of contacting points. An extension of this criterion can be used as proposed by [12]–; g_i and h_i are respectively the equality and inequality constraints describing the i^{th} contact –basically, they force a point and a frame of a link to correspond to a point and a frame of the environment.

The optimization criterion f is optional. It allows the user to control the overall look of the obtained posture. For example, the user may want to have human-like postures. In [4], we used the distance to a reference posture, in [5], the potential function f of the planner level is used as optimization criterion.

To solve such an optimization problem we use FSQP (see [13]) which allows the use of any function provided it is at least twice continuous. In particular, it copes with non-linear criteria and constraints.

III. Adding Task Constraints to the Contact Planner

The Posture Generator is a powerful module of our planner. It is used for projections on sub-manifolds of the configuration space as several other methods (see [14] and [15]), but in a very meaningful way, since the user has a great control on the projection method, by defining both the constraints and the optimization criteria. So far, we did not use its full potential during planning : we restricted the constraints to express contacts, stability, robot limits and collision avoidance. Yet functions g_i and h_i defined in (6) and (7) can be of any kind, provided they are twice continuous. We can thus write many other types of constraints than those of contacts C_i.

Equality constraints $g_j(\mathbf{q}) = 0,\ j \in [1, j_i]$ define a sub-manifold on which the posture we are looking for must be. Such a set is called *task* in [14]. It is indeed really close to the notion of task function defined by Samson in [16], which relates to regulating the distance to a sub-manifold. This notion of regulation is the only difference between both definition. This difference encompasses and allows to take into account the discrepancy between the planning in a perfect world with supposedly flawless models and the execution of the plan in a real environment.

Yet we are also using inequalities $h_j(\mathbf{q}) \leq 0$ that restrict this sub-manifold to one of its subsets. We call *task*, as a generalization of [14], a set of equalities and inequalities. Contact is a particular kind of task. It is worth having the parallel with Samson's tasks because we can translate easily our tasks to be executed by a stack-of-tasks [8]. With these remarks, we rewrite (6) and (7) in a more general way

$$\begin{cases} g_i(\mathbf{q}) = 0 \quad \forall\ \mathcal{T}_i & (8) \\ h_i(\mathbf{q}) \leq 0 \quad \forall\ \mathcal{T}_i & (9) \end{cases}$$

the $\mathcal{T}_i$ being tasks.
Through the addition of tasks to the Posture Generator, inclusion of tasks may be then done in our planner in a very smooth way which does not need particular coding effort, but the interface to insert these tasks in the posture generation problem related to a new node: for each node, the posture generator will solve the problem 1, with the equations of the tasks $\mathcal{T}_i$ as additional constraints.

Here are some tasks that may be of great interest in a motion planning:

- maintaining the orientation of a carried object. We will give an example later;
- keep looking at a target, or maintaining visual features in the field-of-view. The latter can be useful, e.g. for self-localization;
- keeping in the field-of-view, when possible, the contact that is being created, e.g. once the plan is executed on the real robot, visual feedback is possible, etc.

Let's take as an example the following mission: the robot must carry a glass containing a liquid. If the robot does not have the glass in the gripper, it must first reach it. This part of the planning was solved in [4]. Separating the problem into sub-missions as "go to the glass", "grasp the glass", "carry it", and so on, would be the purpose of an higher-level task planner and is beyond the scope of this work. Here we assume the robot with the glass grasped. In order not to spill a drop of liquid, the robot must keep at all time the glass in a vertical position, and in particular at each witness posture associated to a node of our planner. This task can be described with:

$$\mathcal{T}_{\text{glass}} = \begin{cases} \mathbf{n}(\mathbf{q}).\mathbf{i} = 0 \\ \mathbf{n}(\mathbf{q}).\mathbf{j} = 0 \\ -\mathbf{n}(\mathbf{q}).\mathbf{k} < 0 \end{cases} \quad (10)$$

where $\mathbf{n}(\mathbf{q})$ is the axis of the hand carrying the glass, and $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ is a frame of the world (cf Fig. 2). This task is added to every posture generation. Every posture of the output plan will respect its constraints. Adding a task in the planning decreases the number of degree of freedom of the robot, meaning a witness posture will be found for fewer sets of contacts. It has thus a direct impact on the plan, since it reduces the possibilities of the planner.

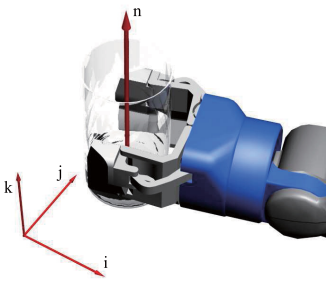

Fig. 2. Vectors linked to the definition of task $\mathcal{T}_{\text{glass}}$

IV. Improved potential field based on global trajectory guide

We present here a refinement in the way we build the potential field guiding the planning. This refinement aims at reducing local minima, making it possible for the planner to tackle more advanced scenarios.

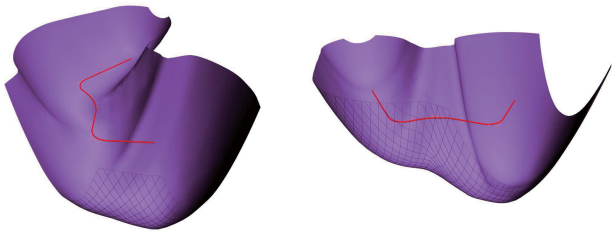

Fig. 3. Based on a trajectory in the configuration space (in red) which acts as a trajectory of reference postures, we build a descending valley-like potential by combining the distance to the trajectory and the distance traveled along the trajectory.

In potential field based planning local minima are a major cause of inefficiency, even if there are methods to escape from them. In classical potential field based planning (see [10]), local minima arise from the repulsive fields of the obstacles. In contact point planning however, obstacles are not to be avoided because the robot may need them as supports (even the floor is then considered as an obstacle in the workspace). We can not have potential fields that drive the robot away from them. We only have an attractive potential function toward the goal. Yet while evolving in this potential field, the robot may be blocked by some obstacles which *de facto* create local minima (just as inequality constraints in optimization), e.g. when there is an obstacle in the steepest descent direction. These are often huge local minima for which practical escape is impossible. To avoid these minima, we use a rough trajectory T, that acts as a guide to go around

obstacle, as well as giving information about the preferred posture at each location.

Upon this guide, we construct a descending valley-like potential field over the configuration space, as depicted in Fig. 3 and explained in [5]. Roughly speaking this potential field embed two terms: the distance d from the current configuration to the trajectory T, and a measure of the distance l traveled along the trajectory.

$$f_T(\mathbf{q}) = d(\mathbf{q}, T)^2 - k \cdot l(\mathbf{q}) \tag{11}$$

T thus acts as trajectory of reference postures. While this copes with relatively big local minima and proved to be effective, we do not tackle some smaller ones, that are linked to the intrinsic discrete nature of contact planning: advancing along the rough trajectory may require to move lightly and/or shorty away from it, what is discouraged by the shape of f_T, and thus wont be the first option considered by the planner.

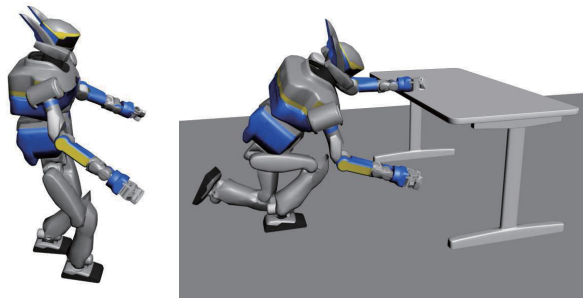

Fig. 4. Two cases where advancing forces moving away from the reference posture.

Fig. 4 depicts two of these cases. Making a step as on the left figure requires to move away from a straight stand-up posture. The second term of eq. (11) however helps to make steps, but these steps are small. When the robot is in a four-legged posture the steps are even smaller because for the same angle change as in an upright posture, the contacting spot moves less. Yet the robot moves forward. The second case (right) is much more an issue: in this case the robot wants to pass under the table. To do so, it has to move from an upright posture to a four-legged one. The rough trajectory T indicates this. But passing from one type of posture to the other requires to use intermediate postures that are neither close to the upright posture nor to the four-legged one. This builds a potential barrier in f_T before which stands an important local minimum, that may only be overcome at high price, timewise. More generally, passing between two highly different postures will lead to such barrier. If postures are close, like sitting and standing-up postures, this difficulty is not encountered.

In both cases, the problem can be interpreted as a lack of anticipation of what lies "ahead". Anticipation would push going over potential barriers. We propose as a solution to use similar potential fields as our global field f_T, but acting only on some bodies of the robot: from a global trajectory T we can derive the trajectories T_i of each robot's body and define a potential field $f_{T_i}(\mathbf{q}) = d_i(\mathbf{q}, T_i)^2 - k_i \cdot l_i(\mathbf{q})$ based on it. We then build a total potential field

$$f'_T(\mathbf{q}) = \alpha f_T(\mathbf{q}) + \sum \lambda_i f_{T_i}(\mathbf{q}) \tag{12}$$

where the λ_i are parameters whose values are 1 or 0 depending on whether we take into account the i^{th} individual potential field or not. Typically we take $\lambda_i = 1$ for the feet and hands and sometimes for the knees. This methods allows the selected bodies to anticipate by being ahead of the overall motion: since advancing along a trajectory gives a bonus, this compensate for a biggest yet meaningful gap between the posture found and the reference posture given by T. Effectiveness of this approach is shown hereafter.

V. Simulation examples

In [5]), we demonstrated how the planner copes with heavily constrained spaces. But the robot was beginning inside a tight space. We were thus legitimately asked about the capacity of our planner to drive the robot so as to *enter* such constrained spaces. We choose scenarios here that demonstrate this capacity, which, as we will see, is only effectively achieved with our second contribution.

A. Move and sit on a chair at a table holding a filled glass

Our first scenario aims at demonstrating the inclusion of an additional task in the planning. To do so we take the scenario we describe in section III, namely carrying a filled glass without spilling a drop of liquid, and we merge it with the planning of sitting on a chair at a table. This planning is difficult since it imposes the robot to enter a narrow space.

Planning is successfully achieved in about 4 hours during which 5400 nodes are generated. The output plan consists in 69 nodes, some of which are depicted in Fig. 5. The robot is first walking, then by helping with his left hand it finally manages to place its left foot in front on the chair (around the 50th node). At this stage it has found an entry point into the narrow space between the table and the chair and begins to move on the chair with the help of its thighs. The same scenario using the modified potential field of section IV is done in about 3 hours and 3800 nodes.

B. Passing under a table

Our second scenario emphasizes on the gain due to individual potential fields. Passing under a table requires to cope with both the very small steps of four-legged motion and, above all, the change of posture types, which happens twice: the robot stands at first in front of the table, must then pass under it, and finishes in an upright posture on the other side. The difficulties are illustrated by Fig. 6. Both figures show the free flyer (i.e. waist) position for every valid posture generated during the planning. In the upper figure, no individual potential field is used. The planner has big difficulties to enter and evolve under the table. 20000 nodes are generated before the planner stops unsuccessfully. This explains the huge number of points before the table. On the contrary, with individual fields (lower part) the planner ends successfully in 1.3 hours and 2916 nodes. The result is a

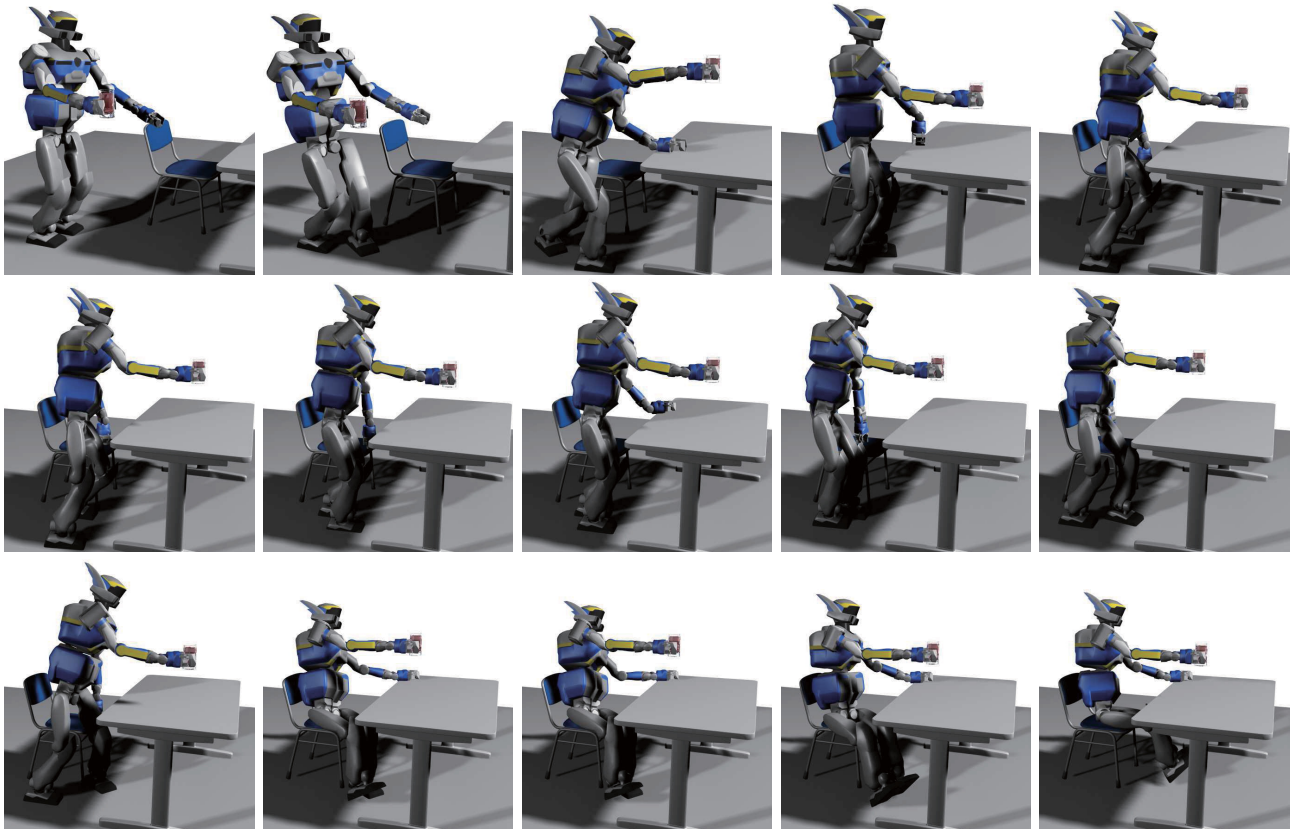

Fig. 5. 69 nodes plan of sitting on a chair while holding a filled glass. Plans is displayed every five nodes.

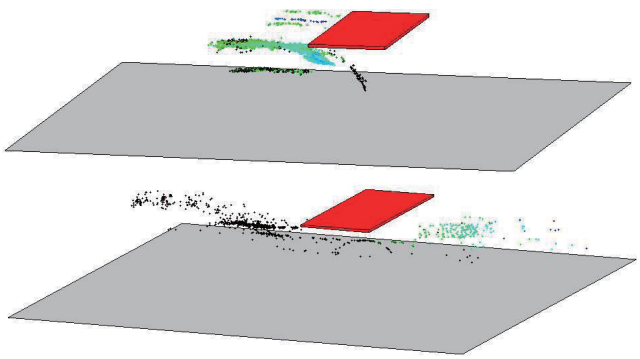

Fig. 6. Position of the free flyer (waist) of the robot for each generated node of the tree built by the planner. Planning without (upper), and with (lower), individual trajectories. For the lower picture, the robot starts lightly earlier by a bit of walk. This correspond to the leftmost points.

141 nodes long path depicted on Fig. 7. Half of the planning time is spent before the table, while passing under the table is done with only 30 nodes.

VI. Experiment cases study with HRP-2

Using the framework described in [5], we played the output of the glass scenario on HRP-2 robot. Since some posture were really demanding for the robot in term of torques, and some steps redundant, we manually removed some nodes. We were able to play successfully the plan with a glass filled with tea. Some shots of this experiment are given by Fig. 8, and the attached video shows the planning workflow and the final trajectory executed by HRP-2.

VII. Conclusion

We presented several extensions to our planner. Namely, how to cope with additional tasks and an improvement of the potential field function we used that allow us to plan faster and in more difficult scenario. Some more works remain. First, the planner tends to propose redundant contacts sets, or contacts that are too close. Second, it would be cleaner to take torque limitations into account at the contact planning level, whereas they are currently dealt with during the post-processing.

References

[1] K. Harada, H. Hirukawa, F. Kanehiro, K. Fujiwara, K. Kaneko, S. Kajita, and M. Nakamura, "Dynamical balance of a humanoid robot grasping and environment," in *IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004, pp. 1167–1173.

[2] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *IEEE/RSJ International Conference on Humanoid Robots*, December 5-7 2005, pp. 7–12.

[3] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[4] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on HRP-2," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, Beijing, China, October 2006, pp. 2974–2979.

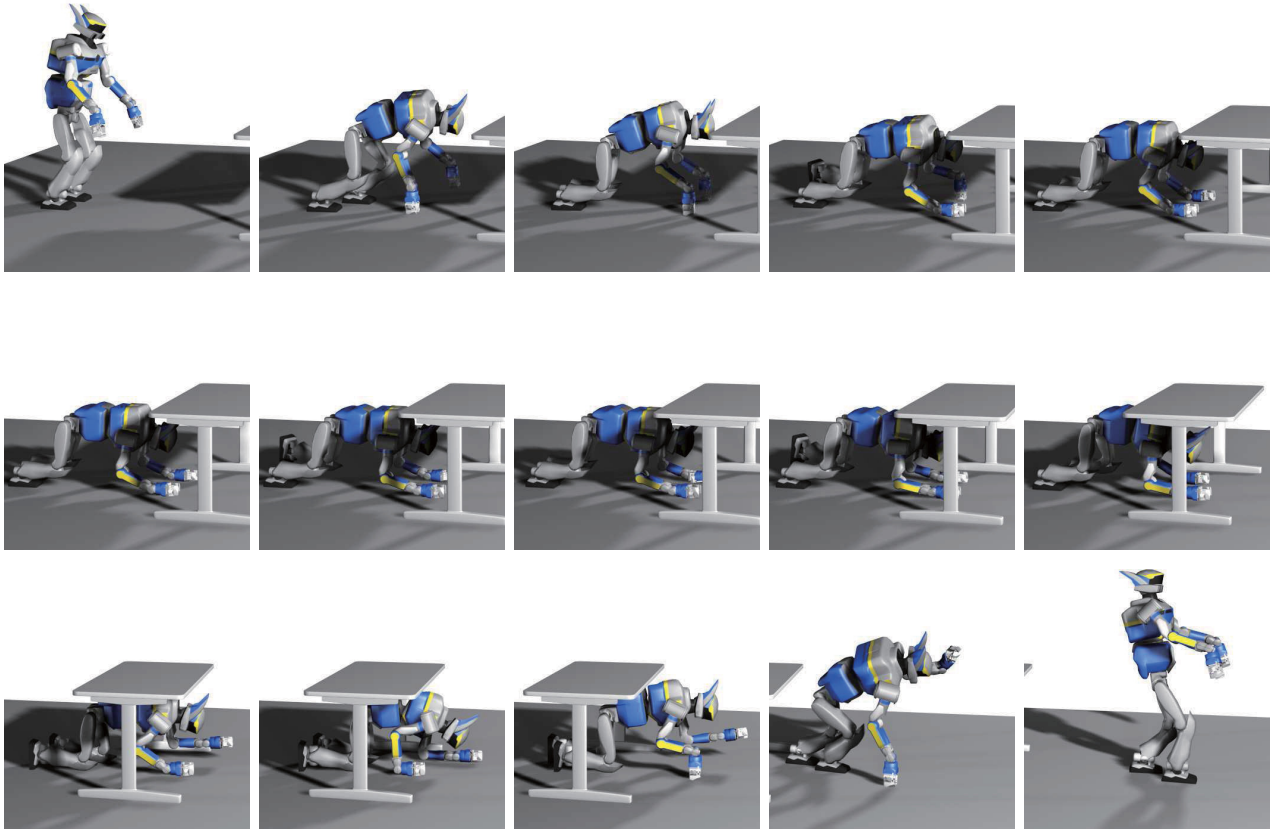

Fig. 7. Passing under the table using individual trajectories for hands, feet and knees. The total movement is made in 141 nodes (from 0 to 140). Between two pictures, there is a "time" of 10 nodes. Note that the longest part of the plan is to go from the upright posture to the four-legged one.

Fig. 8. The glass scenario played on HRP-2. The glass remains perfectly vertical during the whole motion.

[5] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning support contact-points for acyclic motion and experiments on HRP-2," in *International Symposium of Experimental Robotics*, 2008.

[6] A. Kheddar and A. Escande, "Challenges in contact-support planning for acyclic motion of humanoids and androids," in *International Symposium on Robotics*, 2008.

[7] 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 *IEEE International Conference on Robotics and Biomimetics*, 2006.

[8] N. Mansard and F. Chaumette, "Task sequencing for sensor-based control," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, February 2007.

[9] O. Khatib, L. Sentis, and J.-H. Park, "A unified framework for whole-body humanoid robot control with multiple constraints and contacts," in *European Robotics Symposium (EURON)*, ser. STAR Series, S. T. in Advanced Robotics, Ed., Prague, Czech Republic, March 2008. 2008.

[10] J.-C. Latombe, *Robot motion planning*. Boston-Dordrecht-London: Kluwer Academic Publishers, 1991.

[11] K. Bouyarmane, A. Escande, F. Lamiraux, and A. Kheddar, "Potential field guide for humanoid multicontacts acyclic motion planning," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, 12-17 May 2009.

[12] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, pp. 794–807, August 2008.

[13] C. Lawrence, J. L. Zhou, and A. L. Tits, "User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints," 1997.

[14] M. Stilman, "Task constrained motion planning in robot joint space," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, 2007.

[15] J. Cortés, "Motion planning algorithms for general closed-chain mechanisms," Ph.D. dissertation, 2003.

[16] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.