Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2

Adrien Escande, Abderrahmane Kheddar, Sylvain Miossec, Sylvain Garsault

Planning support contact-points for acyclic motions and experiments on HRP-2

Adrien Escande, Abderrahmane Kheddar, Sylvain Miossec, and Sylvain Garsault

AIST/CNRS Joint japanese-french Robotic Laboratory (JRL)
adrien(dot)escande(at)ensmp.fr, kheddar(at)ieee.org,
sylvain(dot)miossec(at)aist.go.jp, sylvain(dot)garsault(at)aist.go.jp

Summary. This paper presents improvements in contact-before-motion planning for humanoid robots that enables to find a path in very constrained environments. Starting from our previous work, the main novelties are to use a rough trajectory to drive the search and as a criterion to find new contacts and generate the best nodes first. This way only few nodes are effectively explored, speeding up the planning process. We experience the algorithm on the real humanoid HRP-2 in a complex scenario.

1 Introduction

When planning a path for humanoid robots, the preferred locomotion mode is walking, with two main ways to do it: footsteps planning [1] or cyclic legged motion along a path planned for a reduced model of the robots [2]. These approaches are effective in environments with flat or stairs-like floor such as offices when clearance is enough. They can be further adapted to use the remaining freedom of the robot to perform some extra tasks. However, they are not designed to cope with environment where the floor is complex or space is tight, or more generally where the environment requires an acyclic motion, that may involve contact supports from all parts of the robot. Such is the case of a robot evolving in an unstructured environment, but is also already the case in human-friendly environments containing a ladder, a table under which a task is to be performed, etc. Planning such motions has recently been addressed by the mean of so-called contact-before-motion planning [3] [4] [5]. Search is done in the contacts space first, and then a path is deduced in the configuration space. Contact-before-motion planning has already exhibited rather good simulation results in scenarios where the workspace was not much constrained or support contact point candidates were already given. We propose here an extension of our previous work that tackles much more constrained environments. We demonstrate our results with a challenging sce-

nario for a humanoid that is yet inspired from everyday life: going out from a chair when sitting at a table.

2 Technical approach

2.1 Definition

We name *contact* the match between one point and frame of the robot's body with one point and frame of the environment such as the distance between the body and the environment is 0. We are mostly using plane-plane contacts to encompass stability. In this case, points to be matched are taken on the planes and frames are defined by a normal vector to the planes (inner for the body, outer for the environment) and a vector of the plane.

2.2 Overall planner

In [3] 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. 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 is inspired by the potential-field-based planner Best First Planning [6]. However, we are planning here in SC. This allows a dramatic reduction of the search space compared to the usual configuration space, but 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 one 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.
For a given set of contacts as input, the posture generator writes and attempts to solve an optimization problem with (non-linear) constraints, whose variables are the degrees of freedom of the robot. Constraints include the matches of points and frames of the contacts, collisions with environment and self-collisions, joint limits and stability. A successful output is a configuration of the robot which respects these constraints. Upon failure of the posture generator, the set of contacts is discarded. The optimization criterion is optional. It lets the user control the overall look of the obtained posture. For example, the user may want to have human-like postures. In [3], we used the distance to a reference posture.
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 postures. It must be noted that associated postures are only feasibility witnesses of the sets and can be changed by post-processing.

2.3 Set of contacts generation

For a given set of contacts, sons are built by either adding a new contact or deleting an existing one.
When deleting a contact, we keep it as geometrical constraints for the posture generation, but it does not participate in the stability of the robot any longer. With this conservative way, we check for feasibility at the exact moment the contact is broken, in particular, we check that the removed contact is in the stability area of the remaining one(s). This makes it extremely likely to find a feasible path between the postures of the old and new sets of contacts. When building sons, the planner tries to delete every existing contact in turn. Generation of new contacts is one of the most difficult part of contact planners and an efficient solution is the major theoretical contribution of this paper. We first discuss the basic challenges and solutions. New contacts must be as few as possible to avoid a combinatorial explosion, yet numerous enough to correctly cover the possibilities of the robot. Moreover, contact candidates (i.e. before the feasibility check by the posture generator) should lead to a failure of the posture generation as seldom as possible, since failure is most of the time much more time consumming than success. For new contacts, we also use a similar heuristic as with removed contacts: a set with a new contact is checked without this contact being taken into account in the stability constraints. The new contact must be in the feasibility area of the old one(s). In [3], contact candidates are generated this way: we predefined contact spots on the robot by means of points and frames attached to bodies of the robot as well as a convex contact surface for each of them. We also give the flat surfaces of the environment on which we allow contact. Then, for each pair (contact spot, environment surface), we randomly chose several time a point and a frame of the environment surface. This reduces to the choice of three parameters (x,y,θ): a point on the surface, and a rotation around the normal vector of the surface. Although this method works, it does not meet the above quality requirements of contact candidates, ending up with too large computation time in complex scenarios. We present here several refinements.

2.4 Technical improvements

Refinements of the above planner are threefold and detailed hereafter:

- formalize the search space of a new contact and use the posture optimization criteria to determine the candidates,
- base the potential field on a rough trajectory in the configuration space,
- use this potential field as optimization criteria so as to generate best nodes first and speed up the planning.

2.5 Contact search space

A pair (R, E) made of a contact spot and a surface of the environment defines only a floating contact (for example a hand on the table) in the sense that it does not fix the relative position (x, y) and orientation (θ) between the two surfaces. A particular fixed contact between R and E is a choice of these three parameters. The possible contacts for a given pair (R,E) is thus a bounded subset C of a 3-dimensional space. If the surface is convex, so is C.

2.6 Contact candidates generation

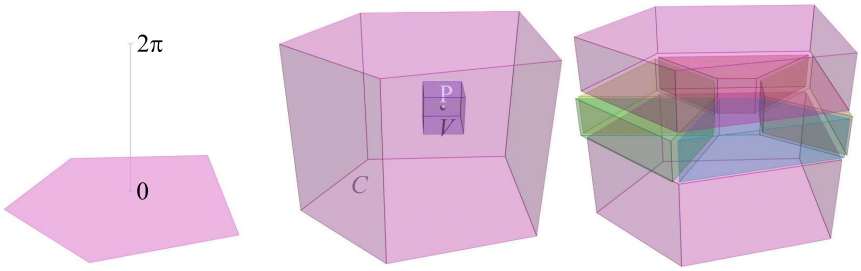

Fig. 1. Search volumes. The environment contact surface (left), the search volume C with a contact point P and a volume V around to be subtracted (middle), the convex subdivision of C/V in C_i(right)

Generating new contacts for a pair (R, E) becomes choosing points in C with the following constraints: (i)points must not be too close, (ii) must ensure a good coverage of C, (iii) must correspond to feasible contacts. Moreover, the process must be fast. The issue (iii) is the bottleneck because feasibility is checked by a call to the posture generator which takes time when it fails. Explicit discretization of C or taking random points in it (respecting (i) and (ii)) is thus not the best idea, since many points may be infeasible. Instead, we let the posture generator decide the point so that (iii) is directly verified if such a point exists. (x,y,θ) is thus chosen according to the optimization criteria of the posture generation. Once a point is chosen, a small volume around it is subtracted to C and a new posture generation is done within this updated C. The process is repeated until either no more point is found or C is empty. Practically, the surface is chosen convex so that C is convex (or cut into convex parts otherwise). When a volume V is subtracted to C, C/V is cut into convex parts C_i on which the search process is repeated. Keeping convex search volumes is important to avoid local minima in the posture generation process.
Following this scheme, we respect constraints (i) to (iii). Furthermore, for each

C_i, we obtain the best node in it. This will prove to be a useful property for further improvement of the new contacts generation.
Contact generation is thus made according to the following scheme:

- for *each contact spot* R
  - for *each surface of the environment* E
    - *Construct the search volume* $C = E \times]-\pi, \pi]$
    - $L \leftarrow C$
    - while L *is not empty, take one search volume* C_{i0} *in* L
      - *attempt to generate a contact in* C_{i0}*. Upon failure skip the two next steps.*
      - *build a volume* V *around this contact*
      - *cut* C_{i0}/V *into convex parts and put it in* L
    - end while
  - end for
- end for

2.7 Rough trajectory

Simple potential functions f as the distance to a goal are enough in simple environment or when there is no big obstacles between the initial position and the goal. However it gives poor results in the other cases, because it leads the planner into local minima from which the time to escape is huge. Moreover, it may drive the planner along complicated and unexpected path. For example, in the early attempts to solve the scenario presented in 3, the robot was climbing on the table. To overcome the problems of too simple potential function and give the user a way to act on the overall look of the solution path, we build our potential function on a very rough trajectory T in the configuration space. This trajectory is defined by a few key postures. Path between them is obtained by linear interpolation so that we have a piecewise linear curve in the configuration space. This trajectory might be colliding with the environment, even at key postures. As long as the penetration depth is not too big, it will not cause any problem. Its purpose is to give a rough approximation of the intended path in the workspace, together with an idea of the postures along it.
We designed the potential function as a descending valley-like potential field around the trajectory T (see Fig. 2) whose minimum is at the end of T:
we denote F the 6-dimensional subspace of the configuration space corresponding to the translation and rotation of the base frame of the robot (free flyer). Let P be a point of the configuration space, P' and T' are the projection of P and T onto F. We suppose that the projection p from T to F is bijective, that is to say no points of T have the same translation and rotation coordinates. This is a reasonable hypothesis in our case. We denote Q' the closest point of T' to P'. Let $Q = p^{-1}(Q')$ and s' the curvilinear abscissa of Q' along T'. We then define the potential function f_T base on T as:

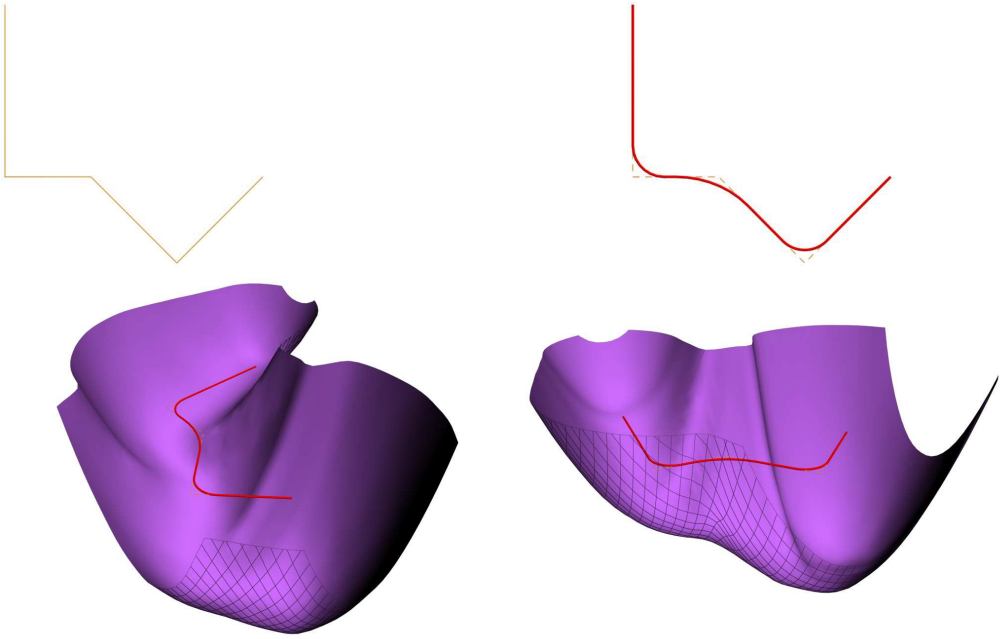

Fig. 2. A rough trajectory and the potential field based on it.
A piecewise linear curve (upper left) is smoothed around its corners (upper right), before a valley-like potential field is built on it (lower left and right)

$$f_T(P) = \|P - Q\|_2^2 - k \cdot s' \tag{1}$$

where $\|.\|_2$ is the euclidean norm and k is a positive coefficient.
The first part of this function aims at keeping the planner as close as possible to the rough trajectory and gives the function its valley-like shape, the second part introduces a slope that will drive the planner towards the goal by measuring the path made along T. k is a weight that controls the relative effects of these two parts.
Because we will need f_T to be a two times continuous function of P for the heuristic of the next section, we needed to slightly reshape T. Discontinuities of derivatives happen when Q' jumps from a segment of T' to another one. There are methods to regularize the distance field to a curve, for example the R-functions [7] [8], but they do not allow an easy computation or even definition of Q'. We thus chose to replace the corners of T' by part of hypercircles with a radius as big as possible provided the distance to the original curve does not exceed a fixed value (see Fig 2). This ensures the two times continuity as long as P' is closer to the curve than the radius, which is enough for us.

2.8 Using global potential field as local optimization criterion

Up to now, when adding a contact to a set S, the planner generates every possibilities, ending with a set of leaves L_S, that also includes leaves obtained by removing a contact from S. However, we know it will first consider only the best leaf l_0 in L_S. Let's denote $\mathcal{L}$ the set of all existing leaves outside L_S. If l_0 is better than any leaf of $\mathcal{L}$, then the planner will select it and build its sons. These sons will then be compared to the elements of $\mathcal{L}$ and of L_S/l_0, but if we know which leaf l_1 is the second best one in L_S then we only need to compare them to the elements of $\mathcal{L}$ and to l_1. Let's go further: we only need to know the best son of l_0 to decide which leaf is the best among all existing leaves (in $\mathcal{L}$ and L_S).
We thus remark that: if we are able to generate the sons of a set in the decreasing order of their score with respect to the potential function f_T, then we only need to maintain one lef among its sons at all time (when a leaf is selected, it becomes a node). That is, we could only generate one son at first, generate a second one when the first is selected, and so on... We would end up with a tree for which each node has a single son that is a leaf.

The advantage of implementing this remark is huge: we need to generate far fewer leaves, and since we do not generate until it is not possible, we have far less costly failures of the posture generator. We can roughly expect to be 10 to 20 times faster. But the remark is based on a strong hypothesis: the capacity to generate sons in the good order. We do not fully have this capacity: it is really difficult to predict for which pair (R,E) or which removed contact we will have the best leave. However for a specific pair (R,E), we already know that we can generate sons in the decreasing order of their optimization criteria value. By using the potential field f_T as optimization criterion for the posture generation, we can thus generate sons in the good order with respect to f_T, for each pair (R,E). We needed to regularize f_T because the posture generator is using two times continuous functions. We end up with an hybrid solution where we still have to generate every leaves obtained by deleting a contact, and have to maintain only one leaf per pair (R,E).

With this solution, computation is sped up by a factor 3 to 5 for the kind of scenarios we tackle. It must be noted that this does not affect the complexity of the tree, but it avoids a large number of posture generations.

3 Results

3.1 Getting out of a chair scenario

To demonstrate the capacity of our planner, we set up a scenario as depicted in Fig. 3. The robot is sitting at a table on a fixed chair and is asked to go to a position on the side of the table. It can use its feet, thighs, knees, forearms and hands to do so and can take contact on the table's top surface,

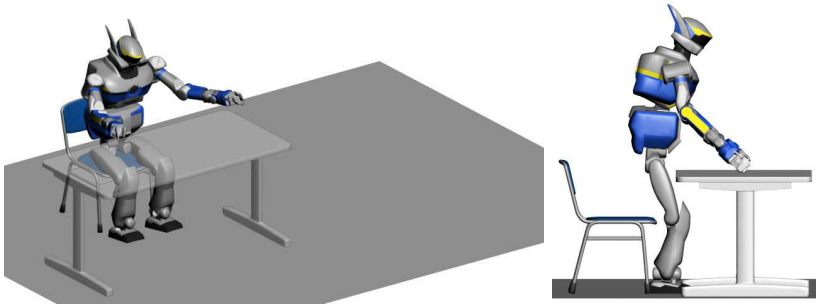

Fig. 3. Getting out of the chair scenario

the horizontal of the chair and the floor. The goal is set as a coordinate the waist must exceed.
The main challenge of this scenario consists in the really narrow space around the chair, and especially between the chair and the table, making it really difficult for the robot to stand on its sole feet. There are numerous potential collisions involving the legs, from self-collisions to the collisions with the legs of the chair. The fact the chair cannot move is both a limitation of our planner, which does not handle moving objects, and an additional challenge for it, since it can not widen the moving space of the robot as a human would do.

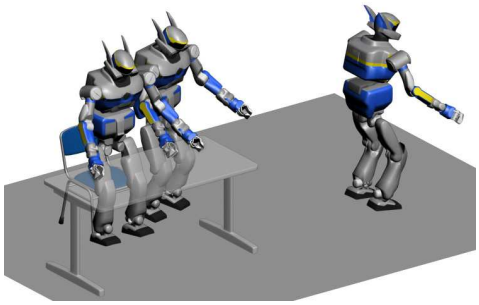

Fig. 4. Key postures defining the rough trajectory T. The robot is slightly floating above the floor and collides with the chair during the interpolation.

The overall trajectory T is defined by four key postures, depicted in Fig. 3 (left) and Fig. 4

3.2 Output plan

The output of our planner for this scenario is a plan consisting in 81 sets of contacts together with their associated witness postures. Fig. 5 displays a

chosen subset of the witness postures that illustrates how the robot makes use of its hand to help him to go outside of the chair. The robot needs 19 nodes to put a first foot on the chair side, and 22 more to have both feet outside of the chair. To do so, it needs to help with its left wrist on the table, then its right hand on the chair. The remaining part of the movement is a static walk. The total movement corresponds to a sequence of 30 steps which, while being small, do not differ much from what a human would do.
The planning took around 5 hours on a PIV 3.4 GHz with 2 Go of RAM. Although it seems to be a long time, it is low compared to the complexity and size of the scene. For this, only 2400 nodes were generated. This is about four times less than without using the method of section 2.8.

3.3 Trajectory between sets of contact

The output of the planner is only key postures and contact data. To have a full path in the configuration space between the initial configuration and the final posture, we need to generate trajectories between two consecutive sets of contacts.
We generate a statically stable trajectory between two sets of contacts S_1 and S_2 (with the witness postures q_1 and q_2) with the following scheme:

- if a contact is geometrically created, broken or changed:
  - a trajectory is automatically generated in the cartesian and orientation space for the involved body (at most one contact changes at a time),
  - the trajectory is discretized in $n-1$ steps, giving n position and orientation constraints C_i, that can be seen as contact constraints,
  - we run n times the posture generator for the contacts in $S_1 \bigcap S_2$ and C_i, $i \in [1, n]$. The optimization criterion is the square distance to q_i, obtained by linear interpolation between q_1 and q_2 and the all contacts in $S_1 \bigcap S_2$ participating in the stability.
- if not, it means a contact is created or broken only at the stability level. In this case we run n times the posture generator with all contacts in $S_1 = S_2$ participating to the stability. Optimization criteria is also the distance to q_i.

This way we obtain a discretized trajectory in the configuration space whose initial configuration is q_1 and final configuration is q_2, which maintains the fixed contacts and moves the body involved in a changing contact.
Sticking together such trajectories computed for every two following sets of contacts, we end up with the full path in the configuration space, as a piecewise linear path.

Trajectories for bodies that geometrically enters or leaves a contact is generated this way for the translation part: if the body leaves a contact, we first have a short straight line whose direction is the normal vector to the contact surface. If the body enters a contact, the trajectory finishes by a straight line with the normal vector as direction. The rest of the trajectory is

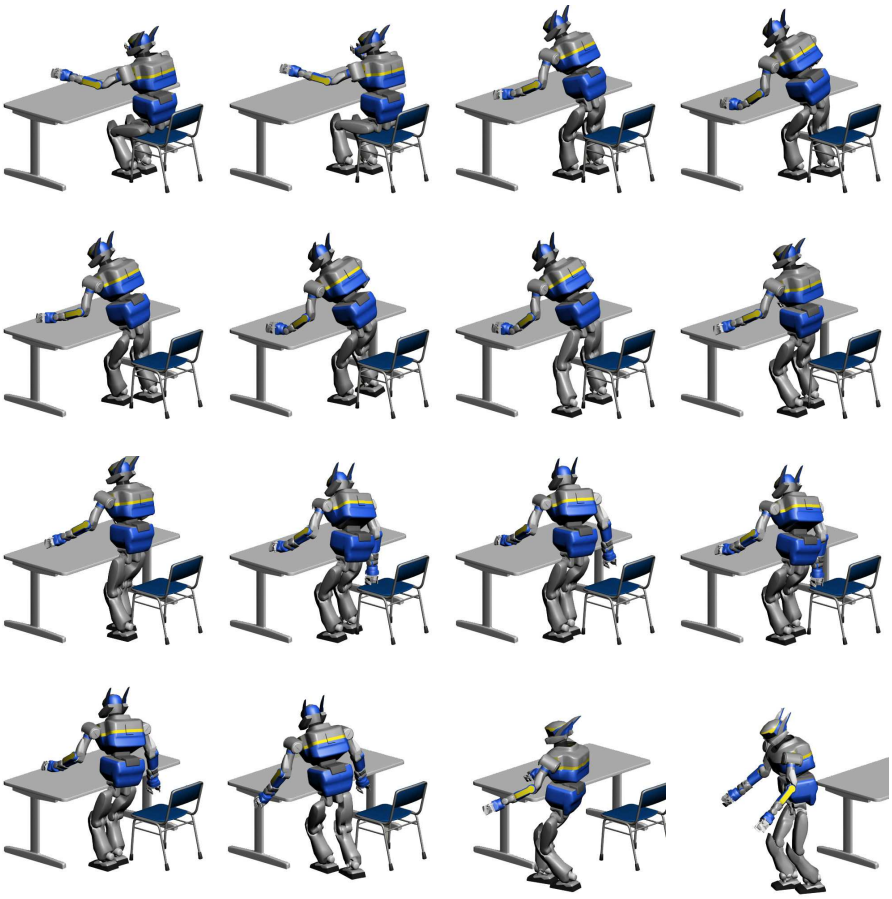

Fig. 5. Some witness postures of the output sequence
The robot is sitting on a chair and move first sightly its feet before getting up, then put its left wrist on the table (first line). It then shift the wrist contact, which allows it to move the left foot outside of the chair (second line). It uses then twice its right hand to help it place its right foot outside of the chair (third line). Finally it can walk toward the goal (fourth line).

a spline. Whenever the spline connects to one of the above straight lines, it is done in a C^1 way. Orientation of the body along this trajectory is obtained as follows: let r_1 and r_2 be the quaternions representing the orientation of the body for the configurations q_1 and q_2. Along a straight line, the body keeps its orientation. Along the spline, its orientation is obtained by spherical linear interpolation between r_1 and r_2.

It might happen that the spline forces the body to be in collision or self-collision. This happened for example when the right hand goes to the chair. In such a case, so far, the spline is manually corrected.

Computing the interpolated trajectory for the whole plan is a matter of about 90 seconds for $n = 20$.

4 Experiments

Once the interpolated trajectory is obtained, it can be executed by the real robot. The scenario replicates the simulated one. The chair, table positions are perfectly known to the robot together with the initial position of HRP-2 to reduce uncertainties and geometric discrepancies. This experiment was reproduced three times to assess its validation and demonstrate the feasibility of whole-body and environment contact support planning in a challenging scenario that has never been carried out in humanoids before. Notice that contrary to some robotic planning problems, these experiments would have been difficult to make even by a full human supervision and interactive instructions on contact stance generation, which shows the sophistication of the method.
To play the path on the real robot, we needed to adapt the speed of the trajectory with the following constraints:

- joint speed must not exceed some maximum speed,
- null speed when creating or deleting a contact,
- overall movement speed of the robot must remain low enough to avoid strong dynamic effects, since the planning is quasi-static,
- yet it must be fast enough to avoid the problem of flowing a high intensity current through the motors for too long.

We used a sinusoidal profile for the speed whose length was controlled according to the above points. Snapshots from getting out from the chair are shown in the Fig. 6 to 9.

This experiment required however several tuning steps that can be avoided by a better use of advanced control scheme. For instance, one of the hard problems we faced was the possibility to switch on and off the stabilizer in case of non-coplanar contacts. The stabilizer is a closed-loop controller acting on the posture of the HRP-2 to correct the effects induced by the flexibility of the ankle. This stabilizer works in connection with the ZMP reference trajectory in walking motion and is necessary to reduce the non-controllable effect of the passive flexibility of the angles. However it has been designed to deal only with planar contacts and must be switched off when this is not the case. Since we allow contact to occur on all the body of HRP-2 with any part of the environment, the stabilizer needs to be switched off when HRP-2 takes support on the table or on the chair. The stabilizer on-off transitions result in discontinuities in the posture configurations. Moreover, for a proper use, one needs to switch it off just before a non-coplanar contact is created and this is difficult to guarantee. In other configuration one needs to switch it on just after the contact is released and only coplanar contacts remain in the

stance, but the abrupt change of posture can create undesirable collision in very constrained space (as is the case here). We could finally find the right timing and tuning by hand, but clearly, a multi-contact stabilizer and dynamic motion generation is to be developed.

Another issue, which we already emphasized in our previous work [3] is haptic sensing. Clearly, it becomes here clearer that such scenarios require contact and force measurement at the contact. Planning whole body contact stance without haptic sensing is like playing navigation motion without vision. Especially, contacts can occur in spots that can not be viewed by a camera. Here also, we notice in previous experiments that flexibility on the robot's cover is more important and safer than any other compliant method. Indeed, compliant cover not only absorbs impacts but also spread the contact area to guarantee a more stable contact formation. We notice that the compliance of the chair cover allowed HRP- 2 a smoother and safer contact stance transitions when its hand was used as support on the chair.

An important lesson we also learned from these experiments deals with uncertainties. It occurred that a contact is not at the exact geometric planned spot, or that the posture leading to the creation of a new support contact does not achieve the desired contact. In the first case, HRP-2 may loose stability because of the early impact or even slip if the posture is an extreme one. In the second case, it often happens that the contact is finally made during the next motion but in this case, the motion appears jerky at the moment the contact is made. These simple cases require in fact playing such motion in guarded modes. However, guarded mode would call for fast postures and trajectories adjustments, if not regeneration. We are working on the matter actively, namely using a mix of the trajectory optimization technique [9] with the elegant formalism of stack of tasks [10] in the operational or kinematics spaces enhanced with guarded motions.

Finally, experiments show that the contact stances can be filtered in the same way a probabilistic road-map is smoothened after the first rough pass of PRM. There are some stances, that are kept in this experiment, which are useless and can be either discarded or merged with the previous or the next stance. We did not tackle this interesting problem for the time being, but is definitely an open issue for future improvement.

5 Conclusion

We presented here several deep improvements of contact planning for the generation of new contacts, the guidance of the planner and the reduction of unneeded computations. These improvements made it possible to use our planner with a challenging scenario where the HRP-2 robot gets out of a chair in front of a table. We then post-processed the output of the planner to play it, with success and repeatability, on a real HRP-2 robot. Experiments underlined a lot of problems to be tackled in the future, either on the software

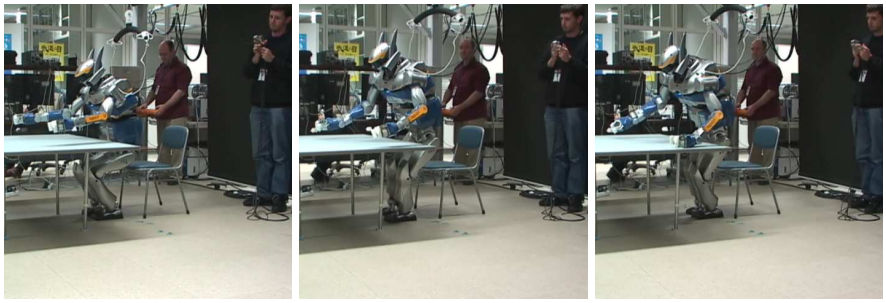

Fig. 6. First part of the experiment: getting up and putting the left wrist on the table

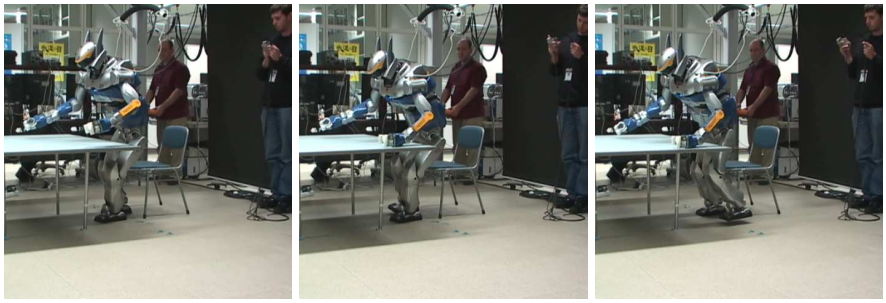

Fig. 7. Second part: moving the wrist contact to be able to move the left foot outside the chair

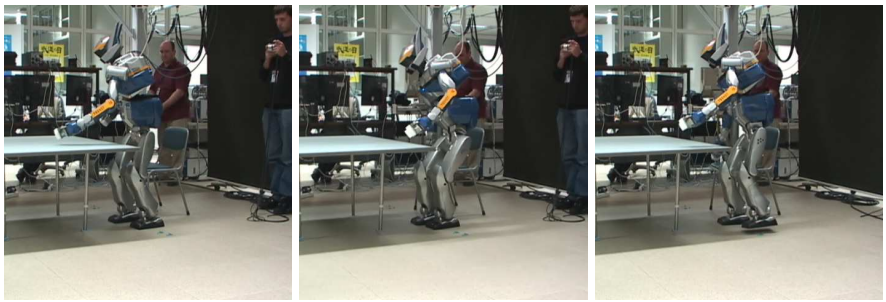

Fig. 8. Third part: moving fully outside the chair with the help of the right hand

level, or even during the design of robots.
From the planning point of view, there is still room for many improvements, finding heuristics to generate fewer unwanted leaves, possibly give the robot a better way to use its environment, for example avoiding to try and put its feet on the chair. As said before, refining the output sequence by deleting some unneeded sets of contacts is a problem we will consider too.

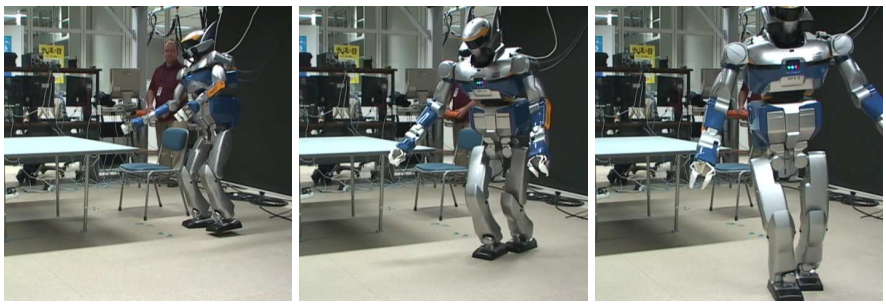

Fig. 9. Last part: moving away from the chair

Rough trajectory is also given by hand up to now. We are working on generating it automatically and trying to make it embed some high level information.

Acknowledgment

The authors would like to thank Kenji Kaneko for his help and insight on the stabilizer issues, Ee Sian Neo and Hajime Saito for their help on various technical aspects of HRP-2 and OpenHRP, and all JRL members who provided useful comments on the experiments. This work is partially supported by grants from the ROBOT@CWE EU CEC project, Contract No. 34002 under the 6th Research program WWW.ROBOT-AT-CWE.EU. The first author is supported by grants from the ImmerSence EU CEC project, Contract No. 27141 WWW.IMMERSENCE.INFO (FET-Presence) under FP6

References

1. Joel Chestnutt, James Kuffner, Koichi Nishiwaki, and Satoshi Kagami. Planning biped navigation strategies in complex environments. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Karlruhe, Germany, October 2003.
2. E. Yoshida, C. Esteves, T. Sakaguchi, J-P. Laumond, and K. Yokoi. Smooth collision avoidance: Practical issues in dynamic humanoid motion. 2006.
3. Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. Planning support contact-points for humanoid robots and experiments on HRP-2. pages 2974–2979, Beijing, China, October 2006.
4. Kris Hauser, Tim Bretl, and Jean-Claude Latombe. Non-gaited humanoid locomotion planning. pages 7–12, December 5-7 2005.
5. Kris Hauser, Tim Bretl, Kensuke Harada, and Jean-Claude Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Workshop on the Algorithmic Foundations of Robotics*, 2006.
6. Jean-Claude Latombe. *Robot motion planning*. Kluwer Academic Publishers, Boston-Dordrecht-London, 1991.

7. Vadim Shapiro and Igor Tsukanov. Implicit functions with guaranteed differential properties. pages 258–269, 1999.
8. Vadim Shapiro. Theory of r-functions and applications: A primer. Technical report, Departement of Computer Science, Cornell University, 1991.
9. Sylvain Miossec, Kazuhito Yokoi, and Abderrahmane 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.
10. Nicolas Mansard and François Chaumette. Task sequencing for sensor-based control. *IEEE Transactions on Robotics*, 23(1):60–72, February 2007.