HAL
open science

Real-time Replanning Using 3D Environment for Humanoid Robot

Léo Baudouin, Nicolas Perrin, Thomas Moulard, Florent Lamiraux, Olivier Stasse, Eiichi Yoshida

To cite this version:

Léo Baudouin, Nicolas Perrin, Thomas Moulard, Florent Lamiraux, Olivier Stasse, et al.. Real-time Replanning Using 3D Environment for Humanoid Robot. IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2011), Oct 2011, Bled, Slovenia. p.584-589, 10.1109/Humanoids.2011.6100844 . hal-00601300

HAL Id: hal-00601300

https://hal.science/hal-00601300

Submitted on 17 Jun 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Real-time Replanning Using 3D Environment for Humanoid Robot

Léo Baudouin, Nicolas Perrin,
Thomas Moulard, Florent Lamiraux
LAAS-CNRS, Université de Toulouse
7, avenue du Colonel Roche
31077 Toulouse cedex 4, France
leo.baudouin@laas.fr
nicolas.perrin@laas.fr
thomas.moulard@laas.fr

Olivier Stasse, Eiichi Yoshida
CNRS-AIST, JRL (Joint Robotics Laboratory),
UMI 3218/CRT,
Intelligent Systems Research Institute,
AIST Central 2, Umezono 1-1-1,
Tsukuba, Ibaraki 305-8568 Japan
olivier.stasse@aist.go.jp
e.yoshida@aist.go.jp

***Abstract*—In this paper, we illustrate experimentally an original real-time replanning scheme and architecture for humanoid robot reactive walking. Based on a dense set of actions, our approach uses a large panel of the humanoid robot capabilities and is particularly well suited for 3D collision avoidance. Indeed A-* approaches becomes difficult in such situation, thus the method demonstrated here relies on RRT. Combined with an approximation of the volume swept by the robot legs while walking, our method is able to cope with 3D obstacles while maintaining real-time computation. We experimentally validate our approach on the robot HRP-2.**

***Index Terms*—motion planning, replanning, humanoid robots, obstacle avoidance, HRP-2.**

I. Introduction

One of the main goals of humanoid robotics is to enable robots to navigate in complex indoor environments that have been designed for humans. These environments have usually a flat floor, and while the space occupied by the upper body of the robot (or human) usually remains relatively free from obstacles, the lower part is often cluttered with obstacles whose position is frequently changed (such as chairs, cables on the floor, etc.). For this reason humanoid robots are better suited for these environments than wheeled robots when they might have no choice but to step over some obstacles, or move in narrow passages. In order to achieve real-time navigation in dynamic environments, humaniod robots need robust and reactive planning capacity of generating precise leg motions in a short amount of time. The dynamic and stability constraints intrinsic to humanoid locomotion make the problem of trajectory planning (and replanning) particularly difficult to solve in real-time.

There have been not so may studies on real-time humanoid motion planning in dynamic environments due to its complexity of the problem. Previous studies set several hypotheses to reduce the complexity to guarantee the real-time operation, for example restricting the obstacles to 2D shapes [1] or simple geometries [2]. Recently interactive 3D navigation by humanoid [3][4] has been reported, but it is for static environments. In this paper, we consider 3D moving obstacles and handle the collision avoidance with the legs in an accurate way, based on fast motion planning with precomputed dense swept volumes [5], whereas only sparse finite footsteps are considered in [3][4]. In the following sections we describe the algorithms and software architecture that enabls real-time planning and replanning with the humanoid robot HRP-2 in an environment where obstacles are sensed through motion capture. In section II we present the global software architecture while in section III we give more details about the different components. Finally we discuss the results of our experiments and conclude in section IV and section V, respectively.

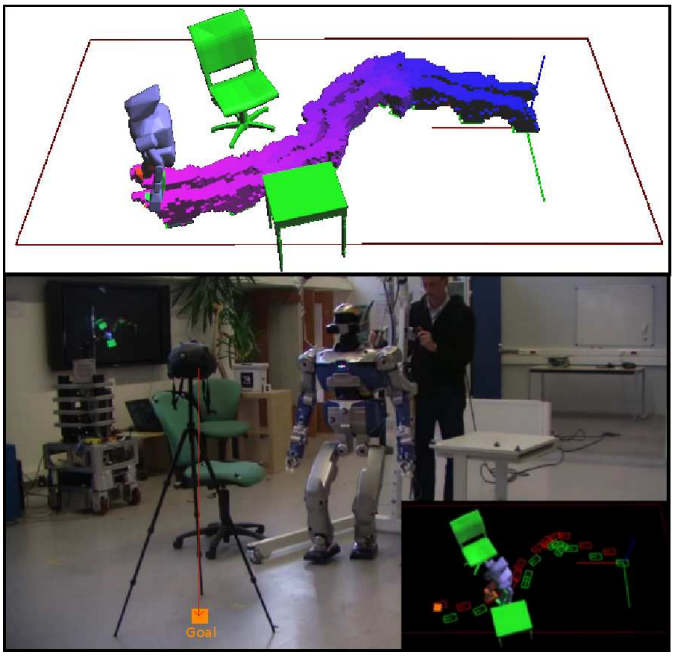

Fig. 1. Top: precomputed swept volumes are used to speed up collision detection for the legs. Bottom: experiment on HRP-2.

II. Global architecture

In this section we describe the global organization of our planning framework. As descibed later, we introduce two distinct parallel processes, “Planner” that plan the trajectory and

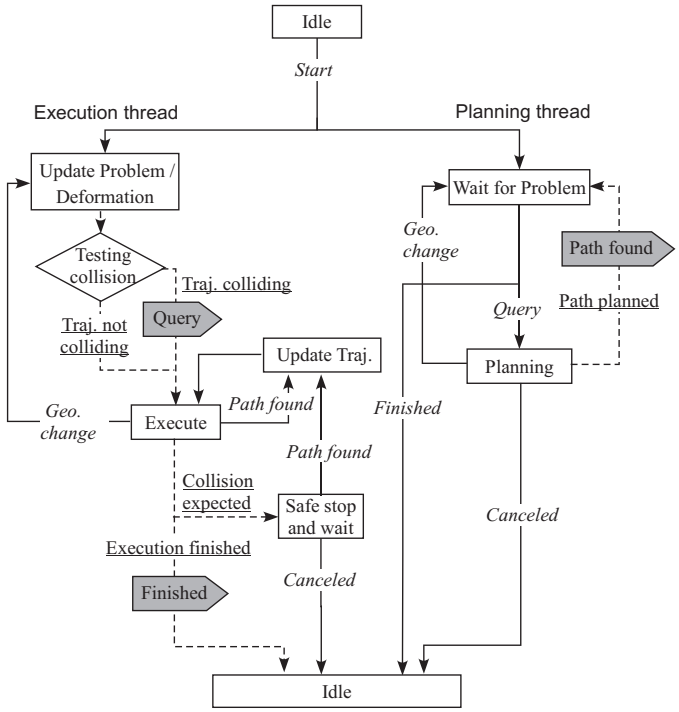

Fig. 2. Replanning process

"Control" that execute the planned trajectory. They communicate with other process such localization system that detects the environmental changes, and the visualization system called "Viewer" to monitor the status of the robots and environments.

As detailed in II-A, the Planner runs continuously to refine the trajectory according to the dynamic environmental changes.

First, we define the confirutation of the robot $\mathbf{q}$ that can be decomposed as follows:

$$\mathbf{q} = [\mathbf{x}, \mathbf{q}_{lower}, \mathbf{q}_{upper}] \in SE(3) \times \mathbb{R}^{6\times 2} \times \mathbb{R}^{18} \quad (1)$$

where $\mathbf{x}$ define the robot spatial coordinates (we call $\mathbf{x}$ the free-flyer), $\mathbf{q}_{lower}$ represents the vector for the joints of both legs, and $\mathbf{q}_{upper}$ is a vector with 18 values for the upper body joints.

The output of the Planner sent to the Control concerns only the lower body ϕ^n_{lower}: this contains lower body joints trajectories as well as the free-flyer $\mathbf{x}$, the CoM (center of mass) and ZMP (zero moment point) trajectories. Here n denotes the trajectory is planned at n-th run of the Planner. In addition, the notation Ψ^N will be used to denote sequences of N footsteps, with SE(2) the rigid motion in a 2D space.

$$\Psi^N \in [SE(2)]^N \quad (2)$$

In order to perform real-time replanning, our global architecture follows the organization illustrated in Fig. 2 which was proposed in [6]. The control loop and the planning are working in independent modules and communicate only when needed through the server (see Fig. 3). In an execution thread, the planned trajectory is sent to the control part, while the

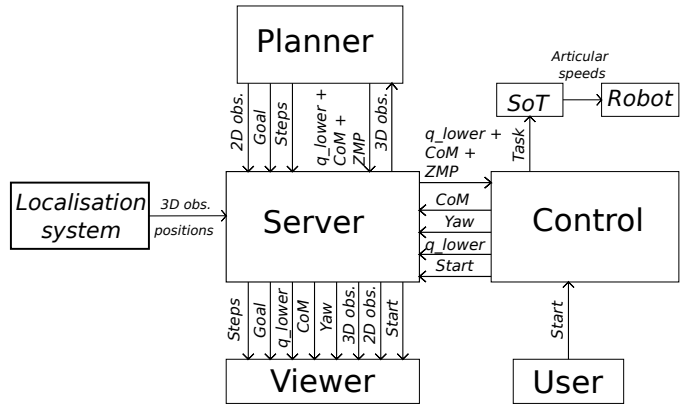

Fig. 3. Connections with the server.

localization information is read to check for potential collision. If a collision is detected along the planned trajectory, a query is sent to trigger replanning.

This organization and the communications through the server are illustrated on Fig. 3. The planner has four data flows. The most important one is the trajectory of the lower body ϕ^n_{lower} that contains leg joint trajectories as well as CoM and ZMP trajectories as defined earlier. The remaining 3 flows are for visualization: 2D obstacles (which are mostly used to simulate artificial obstacles), the current goal position, and the current sequence of steps planned.

The main difficulty for the server is to manage communications between two systems running at different frequency. The control loop reads inputs at a constant rate (for our HRP-2 humanoid it is 5 ms) whereas the planner runs with lower frequency: they are therefore asynchronous. To cope with this issue, the controller uses a buffer to handle the large vectors sent by the planner. This buffer contains a large sequence of $\phi_{lower}(t)$ values to be followed in the future ($t > t_{now}$), and these values are updated every time a new trajectory ϕ^n_{lower} is received. This mechanism is illustrated on Algorithm 1.

Algorithm 1 System control loop

Require: A fixed value $\mathbf{q}_{upper}$ for the upper body joints.
  allocate buffer
  n ← 0
  t ← 0
  loop
    if receive ϕ^n_{lower} **then**
      update buffer
      n ← n + 1
    end if
    $(\mathbf{q}_{lower},\ \mathbf{x},\ CoM,\ ZMP) \leftarrow \phi_{lower}(t)$
    $\mathbf{q} \leftarrow [\mathbf{x}, \mathbf{q}_{lower}, \mathbf{q}_{upper}]$
    apply $\mathbf{q}$ on the robot (using the actuated degrees of freedom)
    t ← t + Δt
  end loop

Algorithm 2 Planning component

loop
  read goal and obstacles positions
  $\Psi_n \leftarrow$ compute new sequence of steps
  if trajectory not totally sent to controller **then**
    create a portion of ϕ^n_{lower} for one step (this portion of trajectory also depends on the next step because of the smoothing process – see section III-A –), and send it to the controller
  end if
end loop

The robot control and stabilization is performed using a Stack of Task ([7]), computing joint references from a set of tasks.

A. Planner

The planner is a loop presented on Algorithm 2. It continuously reads the coordinates of the goal and obstacles, and checks if there is a new target or new collisions. After walking pattern generation (see section III-A), we have a sequence of half steps that have to be inserted in the previous trajectory using paths connection (see section III-D).

After an initial planning, three cases can occur:

- Goal has changed: we replan from two steps after the current one up to the new goal. This value gives to the planner enough time to find a path. It represents a period of about 3.5s.
- Obstacle collides with the current planned trajectory: we either
  - replan from the step before the collision,
  - or replan from two steps after the current one.

  Mixtures between these two strategies have been considered. The new goal is defined as the union of small regions around all the steps occurring after the last collision in the current trajectory.
- Nothing has changed: we take a random initial step *i* (later than 2 steps after the current one) and a random final step *f* in the current trajectory, and replan between *i* and *f*. The goal of this replanning is to find shortcuts in the path (during the planning process if a path is found between *i* and another step than *f* in the current trajectory, we also check whether a shortcut has been found). Newly computed paths are preserved only if they are better than the corresponding sequences in the current trajectory. To compare different sequences we take into account the number of steps (fewer is better) but also favor forward steps with small rotation.

III. Algorithms and Implementation

A. Walking Pattern Generation

When replanning is performed while a humanoid robot walks dynamically, one of the problems that arise is that a step has always some influence over the next step, and as a consequence it is not possible to easily modify the sequences of steps for they are not independent. It is also not always possible to stop the robot at the end of the current step. In [8] was introduced a method for walking pattern generation in which these problems do not occur. The generation of walk sequences is split into two phases. First, we generate a walk with zero speed in the middle and at the end of each step. The generated walk is thus composed of "half-steps" that can be concatenated at will. These half-steps are all dynamic trajectories, but with zero speed connections (i.e. a half-step is a trajectory in the configuration space between two quasi-static poses). It is therefore possible to stop the robot at the end of any half-step, that is to say either in a double support stance or with the swing foot in maximum height position (this height is chosen relatively large to give a good obstacle clearance to the robot). Then comes the second phase during which the half-steps are progressively mixed by creating overlaps between them, which makes them lose their independence. At the end of this phase the walk becomes more fluid, faster and more dynamic. This two-phase approach for walking pattern generation is very convenient for online replanning because we control the independence between half-steps in the sense that the smoothing between two consecutive half-steps can be easily canceled. Thus if for example a sequence of 10 half-steps has been planned and smoothed, it is easy to "break" the sequence after the fifth half-step and replace the next half-steps, and the smoothing need only to be restarted just before the fifth half-step. If we know the time required for the smoothing (which doesn't vary a lot), this replanning operation can be done while the robot performs the sequence of walk.

B. Collision detection

To obtain real-time footstep planning, we need to perform very quick collision checks. To do so we use an approach similar to the one introduced in [5] where swept volume approximations are precomputed offline. The principle follows this remark: after the smoothing process, the final trajectory of one step or half-step of the robot depends on many parameters (smoothing parameters and parameters of the current and next half-steps), *but before the smoothing*, when the sequence of half-step is "raw", the trajectory of the robot locally only depends on the parameters of the current half-step. And half-steps have the good property of needing only three parameters to be completely defined. Because of the low dimensionality of this parameter space, it is possible to obtain a quite dense covering of the set of feasible half-steps with only a limited number of fixed half-steps. So what we do is that we decide a finite set of half-steps in advance (about 200), and we will use only these half-steps to produce sequences of walk. There are two types of half-steps (see [5]): upward and downward half-steps, and roughly for N upward half-steps and N downward half-steps, it is possible to generate $N \times N$ different single steps. Thus, with only a relatively limited number of half-steps we have a large number of available steps, so it gives us a very good expressiveness compared with methods where only 15 to 30 steps are considered [1]. So, once the finite

set of half-steps is chosen, for each of them we create the approximation of the volume swept by the lower part of the robot legs during its execution (see Fig. 1). For the upper part of the robot we are less precise and use rough bounding boxes. In [5], the swept volume approximation are stored in tree structures and designed to be efficiently tested against points of the environment, but in our implementation we use these tree structures to first build triangle meshes representing the swept volume approximations, and then use the PQP algorithm [9] to check collisions against triangle meshes in the environment.

Let us sum up the process again:

- We first plan a collision-free walk sequence walk using only a finite number half-steps (planning phase).
- For each half-step we know a conservative approximation of the volume swept by the robot during this half-step, the approximations are computed offline. Thus, we save a great amount of time during collision checks: indeed, instead of performing many collision checks along the half-step trajectory, we perform just one collision check with the swept volume approximation.
- Once a collision-free sequence of half-steps has been found, we can start to smooth it. However, smoothing the trajectory amounts to a rather unpredictable deformation: we need to do collision checks again. For these collision checks, we use the PQP algorithm with convex representations of the robot bodies (it reduces the number of triangles). Of course verifying a trajectory with these collision checks is slower than with the swept volume approximations, but overall the smoothing process is much faster than the planning phase: indeed smoothing is only done for trajectory optimization and it doesn't need to consider thousands of trajectories like the planning process.

C. Footstep planning

During the planning phase, we use a discretized version of RRT [10] similar to the one used in [5] to search for a collision-free sequence of half-steps. This method is *ad hoc* and not yet completely satisfying as RRT is not very well suited for discrete footstep planning, but since we have about 200 half-steps, this approach is overall better than an A* search whose performances quickly dwindle when the number of possible actions increases.

D. Connection between paths

In several occasions we can have to adjust footsteps in order to connect paths: for example when the planner tries to optimize some parts of the current plan, it uses intermediate goals. And the planning process that tries to reach this goal would decide that the goal is reached not if it is exactly reached, but if a configuration *close enough* to the intermediate goal is reached. In that case, to link the intermediate sequence with the initial one the last half-step must be adjusted. This situation is shown on Fig. 4 where the (half-)step 102 has to be modified. To know how steps can be modified, we use again the property of low dimensionality of the half-steps: offline,

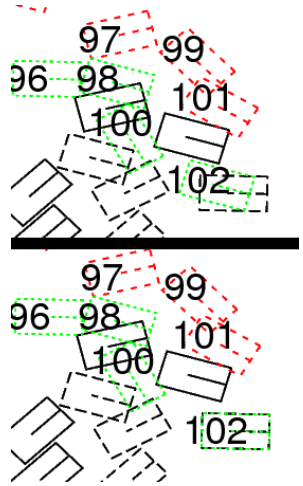

Fig. 4. Connection between two paths at step 102. The last step of the colored path has to be changed to reach the exact position of the old step.

we use extensive simulations to decide an approximation of the continuous region of feasible half-steps (feasible half-steps must avoid self-collisions and joint limit violations). Thanks to the low dimensionality of parameter space for half-steps definition, this construction can be done in a few hours. Since the slightly modified half-steps do not belong to the fixed finite set of half-steps for which we know swept volume approximations, before validating an adjustment we need to check collisions with the environment for the modified half-step.

IV. Experiments

A. Distributed computations

We have conducted experiments with the humanoid robot platform HRP-2. The humanoid robot HRP-2 used in the experiments embeds two computers. One is used for the low-level control and hosts a general inverse kinematics module called Stack of Tasks (SoT in Fig. 3). The other is used here for planning. One remote computer is used to acquire the position of robot, goal and obstacles through motion capture system (for this reason a set of motion capture reflectors must be placed on every potentially moving obstacle). The fourth computer is used to display the current state of the robot and the environment.

B. Environment

All experiments were performed on the robot HRP-2. We used a flat surface of $4\mathrm{m}$ by $6\mathrm{m}$, and obstacles such as a table and a chair.

Obstacles are tracked by the motion capture system, sending 3D positions over the local network. The robot is also tracked through motion capture in order to localize the obstacles relatively to the robot position and orientation. Obstacles are considered to have moved for displacements of more than $2cm$ or more than $5\,\mathrm{deg}$. These values have been chosen to avoid undesirable heavy replanning caused by noise from the motion capture system.

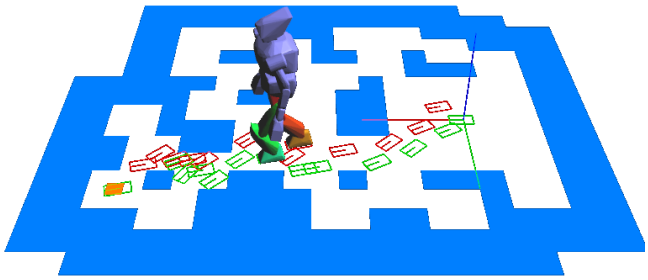

Fig. 5. Simulation of the 2D environment, representing areas where the robot is not allowed to put a foot.

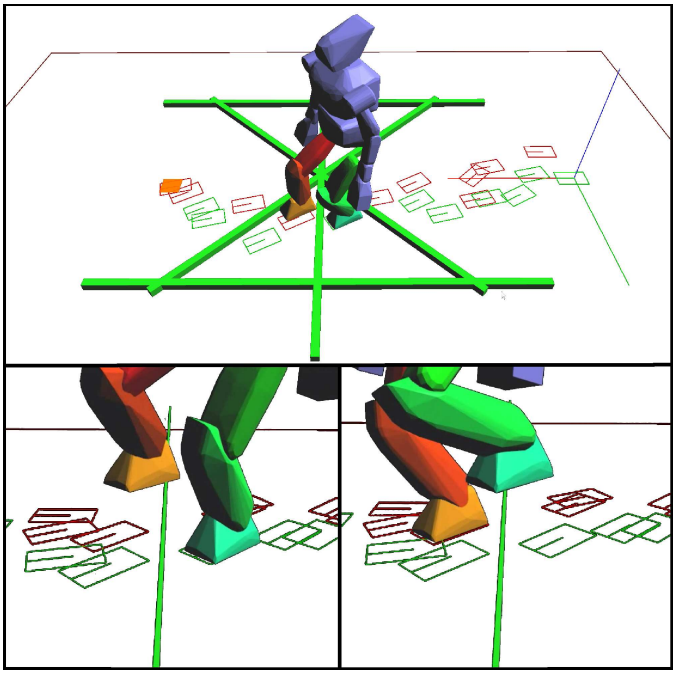

Fig. 6. Top: in simulation, HRP-2 stepping over bars of section 8cm×8cm. Bottom: details on the stepping over motion of an horizontal cable 10cm above the ground.

An object is selected as the 2D goal in the motion capture area (see Fig. 1). This object can move freely and redefines online the new current goal for the robot. The robot behavior is to plan a motion in order to put one of his feet under this object, in a square of side 15cm.

The environment can also have forbidden 2D zones as shown in Fig. 5. The forbidden zones are depicted by colored surfaces. It is also possible to add online 3D obstacles in this kind of environment. However, in this environment, our algorithm is not yet fast enough to find a solution for the whole path in less than the time needed to perform a step. In section V, we propose ideas to accelerate the planning.

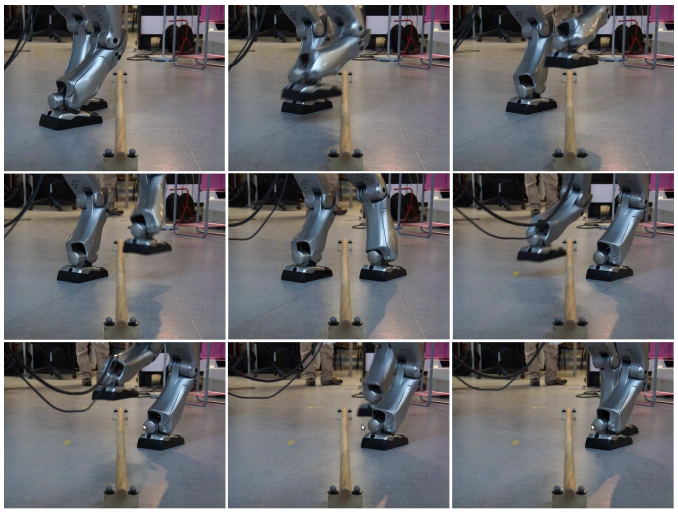

Fig. 7. Details on stepping over a bar placed 8cm over the ground with the HRP-2 robot.

C. Results

The simulations have shown that we can use our planner to step over obstacles as in Fig. 6 where the robot is obliged to overcome bars lying horizontally up to 10 cm above the ground. The smoothing process handles the collision avoidance during the rise and the descent of the foot. In the experiment depicted in Fig. 7, the robot feet pass at about 2cm from the bar.

We tested different scenarios where we modified online the obstacles and goal positions. There are failure cases, but most of the time the planner is able to successfully return new sequences of steps to avoid suddenly expected collisions in the current trajectory. The planner takes about one second to find a path of 4m long with 30 steps, taking into account two obstacles defined with about 60 thousands of triangles in total. For shorter paths (less than one meter long), the planner is quicker, so we have tens of paths in a second. This helps the robot to improve portions of the current path when the obstacles and the goal do not move.

But overall, the planner often does not have time to produce close-to-optimal paths, and the robot occasionally follows unnecessarily long paths. In future work we hope to redure planning time though the use of modern computation technology sush as parallelization.

In the companion video of this paper, we can see that the robot is touching the table. In the current implementation we did not have time to integrate the robot localization and plan accordingly. Therefore the robot plans while following on-line the obstacle position but not with an on-line correction of its own position. However nothing prevent us from doing so, and we believed that this point will be fixed very soon.

V. Conclusions and Future Work

We have presented a real-time footstep replanner for the HRP-2 humanoid robot that uses precomputed swept volumes for fast obstacle avoidance. The RRT algorithm used for

planning does not always return the best path, but it is a good compromise between computation time and the number of steps needed to reach the goal. Our approach is particulary well suited for cluttered environment, and we believe to have obtained preliminary but promising results in our quest towards efficient online replanning of walking motions using non-local methods.

In future work, we will use vision to detect the position of the objects around the robot and the position of the robot itself in the environment which will increase the robot autonomy. We also want to improve the performance of our algorithms by using parallelizing them. We could for example launch several different planning algorithms at the same time, and always choose the best trajectory found.

A better management of the expected computation times would also help us to better improve small portions of the current path, when full replanning is not compulsory.

Acknowledgment

This work was supported by RBLINK Project, Contract ANR-08-JCJC-0075-01.

References

[1] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the Honda ASIMO Humanoid," in *IEEE/RAS Int. Conf. on Robotics and Automation*, 2005, pp. 631–636.

[2] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3D Perception and Environment Map Generation for Humanoid Robot Navigation," *International Journal of Robotics Research*, vol. 27, pp. 1117–1134, 2008.

[3] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "Interactive Control of Humanoid Navigation," pp. 3519–3524, 2011.

[4] J. Chestnutt, *Motion Planning for Humanoid Robots*. Springer-Verlag, 2010, ch. Navigation and gait planning, pp. 1–28.

[5] N. Perrin, O. Stasse, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, 2001, conditionnally accepted.

[6] E. Yoshida and F. Kanehiro, "Reactive Robot Motion using Path Replanning and Deformation," in *IEEE/RAS Int. Conf. on Robotics and Automation*, 2011, pp. 5457–5462.

[7] N. Mansard and F. Chaumette, "Task Sequencing for Sensor-Based Control," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, February 2007.

[8] N. Perrin, O. Stasse, F. Lamiraux, and E. Yoshida, "A biped walking pattern generator based on half-steps for dimensionality reduction," in *IEEE/RAS Int. Conf. on Robotics and Automation*, 2011, pp. 1270–1275.

[9] E. Larsen, S. Gootschalk, M. Lin, and D. Manocha, "Fast Proximity Queries with Swept Sphere Volumes," in *IEEE/RAS Int. Conf. on Robotics and Automation*, 2000, pp. 3719–3726.

[10] S. LaValle and J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Workshop on the Algorithmic Foundations of Robotics*, 2000, pp. 293–308.