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 body 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, humanoid 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 the 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 enables 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.

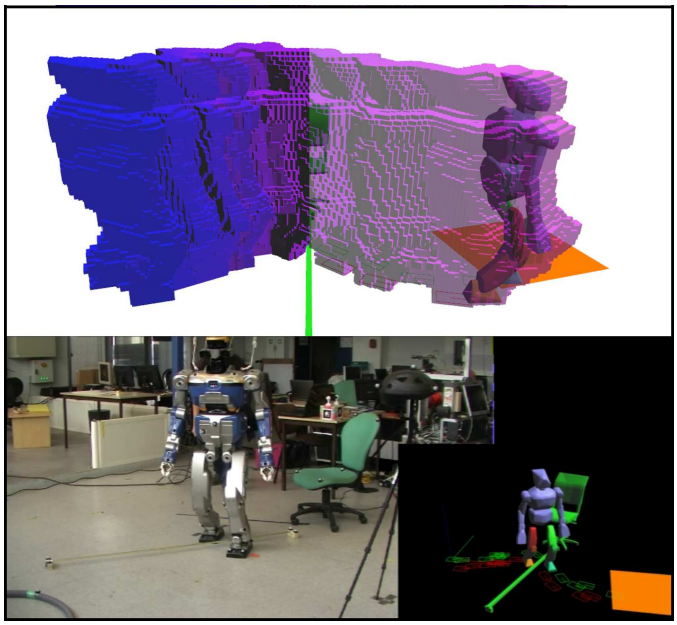

Fig. 1. Top: precomputed swept volumes are used to speed up collision detection for the body. Bottom: experiment on HRP-2.

II. Global architecture

In this section we describe the global organization of our planning framework. As described later, we introduce two distinct parallel processes, “Planner” that plan the trajectory and “Control” that execute the planned trajectory. They communicate with other process such localization system that detects

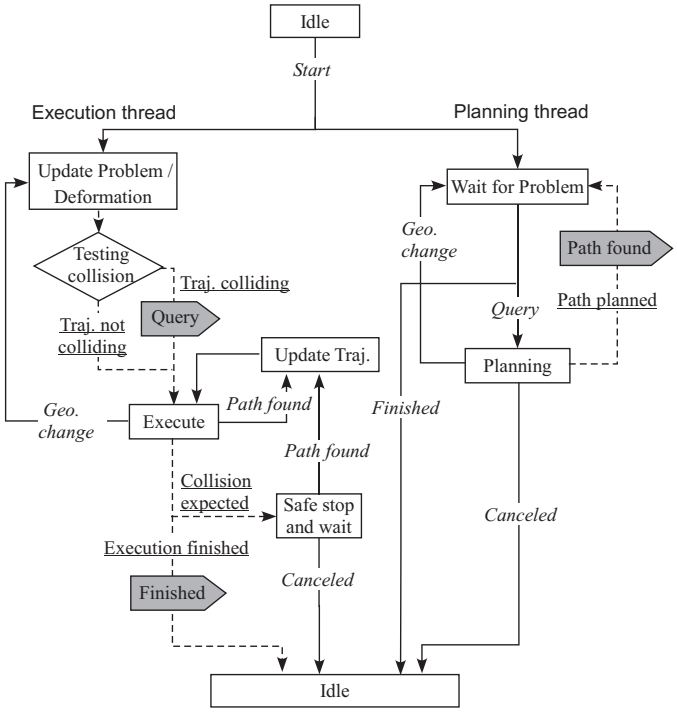

Fig. 2. Replanning process

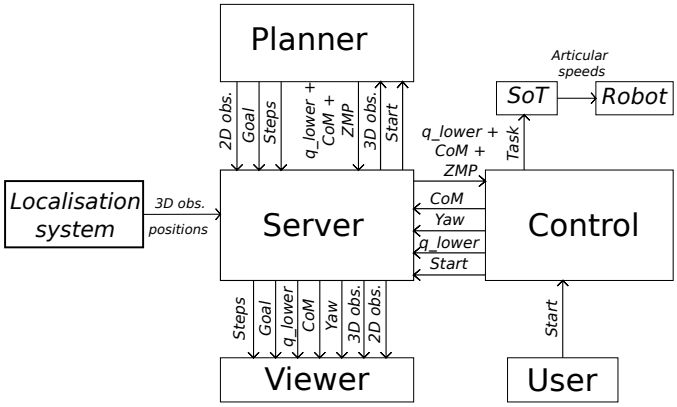

Fig. 3. Connections with the server.

the environmental changes, and the visualization system called "Viewer" to monitor the status of the robots and environments.

A. Motion trajectory and control

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

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

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

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 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.

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

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 localization information is read to check for potential collision.

This organization and the communications through the server are illustrated on Fig. 3. The planner has six data flows. The three most important flows are the trajectory of the lower body ϕ^n_{lower}, the position and orientation of 3D obstacles and the start signal. The lower body trajectory 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.

The robot control and stabilization is performed using

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