Planning and Fast Replanning Safe Motions for Humanoid Robots.

Sébastien Lengagne, Nacim Ramdani, *Member, IEEE,* and Philippe Fraisse, *Member, IEEE*

***Abstract*—This paper introduces effective numerical methods for the planning and fast replanning of safe motions to ensure the safety, balance and integrity of humanoid robots over the whole motion duration. Our safe methods do not depend nor are connected to any type of modelling or constraints. To plan safe motions, certain constraints have to be satisfied over a continuous interval of time. Classical methods revert to time-grid discretization, which can be risky for the robot. We introduce a hybrid method for planning safe motions, which combines a classical unsafe method with a verification step that checks constraint violation and computes excess using interval analysis. When the robot meets unexpected situations, it has to replan a new motion, which is often too time-consuming. Hence, we introduce a new method for rapidly replanning safe motions, i.e., in less than 2s CPU time. It computes off-line feasible subsets in the vicinity of safe motions and finds on-line a solution in these subsets without actually computing again the nonlinear constraints. Our methods are validated using the HOAP-3 robot, where the motions are run without any balance controller.**

***Index Terms*—Humanoid robots, Discretization, Inequality constraint, Feasible subset, Interval analysis**

I. INTRODUCTION

The planning of *safe* motions that ensure the safety, balance and integrity of humanoid robots has seldom been investigated, despite recent achievements in motion planning methods. Humanoid robots are complex systems in which the geometric and dynamic 3D models are highly nonlinear, which may have constituted a severe obstacle to the development of methods capable of planning such safe motions. In this paper, we introduce efficient methods for planning and fast replanning safe motions for humanoid robots. In fact, motion planning for humanoid robots covers a broad range of issues, such as the aspects of the digital actors' locomotion [1], generation of kicking motions [2], computation of manipulator robots' trajectory [3] and smoothing of pre-calculated motions [4]. As a result, the availability of a safe motion planning method would have a significant impact in humanoid robotics as it would provide the conditions for challenging new applications for humanoid robots involved as coworkers [5], [6] or serving as assistive robots in the home. Meanwhile, industry is currently investing in more complex structures, such as two-arm robots (Motoman) that share the same environment as human workers, in order to improve process efficiency and quality in mass production. The complexity of two-arm robots suggests that in the next decade, industrial needs for safe motion should be equivalent to the need for safe motion in humanoid robotics today. This complexity limits the reactive capabilities of computation, as well as the motion constraint validity during movement. Moreover, most of the time the control loop in charge of the humanoid robot's equilibrium compensates the trajectory motion errors. A safe motion planning framework with the ability to produce a movement while satisfying the balance constraint would in theory allow balanced open-loop walking. In practice, it would improve the robustness of the humanoid controller. Actually, the stability margin of this controller could be used exclusively for external disturbances or unmodeled dynamics. This is of interest to complex robotic systems because of the need to improve efficiency, accuracy and safety.

To plan a safe motion, one has to check that the constraints that characterize a robot's safety, balance and integrity are indeed satisfied over the whole motion duration. However, because of strong nonlinearity in some constraints, the optimal motions are usually obtained at the price of long computation times. As a consequence, the sought-after optimal motions are often generated off-line and then used as joint reference trajectories. Some planning methods can yield results faster but they often use simplified or reduced models, e.g. Kajita's cart-on-table model [7], Kajita's resolved momentum method [8] or Goswami's angular momentum balance [9]. Then, they have to check a posteriori using simulation software that closed-loop control can indeed ensure that the constraints on joint values or torque limits, feasible inverse kinematics, and equilibrium are satisfied when implemented on the robot. In the sequel, we will focus on motion planning approaches that use complete whole-body models, and consider equality and inequality constraint satisfaction a priori. To deal with either equality or inequality constraints, available planners need time-grid discretization. We showed, in previous works, that the classical time discretization approach is hazardous since it ensures constraint validity only for the considered time instant, without any information about constraint validity between two time-grid instants. Hence, we propose a new method for safe discretization that relies on a time-interval discretization and uses interval analysis to compute constraint extrema over the time intervals, thereby ensuring constraint validity over the whole motion duration [10], [11]. In this paper, we introduce an iterative hybrid method for planning safe motions, which requires scalable computation time similar to that required by classical unsafe motion planning methods. It computes feasible optimal motions using the classical time-grid discretization approach, then checks a posteriori constraint violation over whole time duration using the time-interval discretization and computes the violation magnitude. It then uses the latter to penalize the constraints. The whole planning/verification

S. Lengagne, N. Ramdani and P. Fraisse are with the LIRMM UMR 5506 CNRS, Univ. Montpellier 2, 161 rue Ada, Montpellier, 34392 France.

S. Lengagne was with the DEMAR project-team, INRIA Sophia Antipolis Méditerranée, Nice, France. He is now with the CNRS-AIST JRL, Tsukuba, Japan.

N. Ramdani is with the PRISME, Université Orléans, 63 av. de Lattre de Tassigny, 18020 Bourges, France.

process is then redone until no violation occurs. A proposition is given which proves that the hybrid method converges to a conservative and safe solution. Experiments show that our hybrid approach converges after only a few iterations. In order to validate our hybrid safe motion planning method, we built a database of computed safe motions, which was used on the HOAP-3 robot to track a moving target, while in an open loop, i.e., without any balance control. Finally, it must be noted that our method for planning safe motions is not bound nor connected to a given model or constraint. It is a generic method, that may be used with reduced models as well, provided the model used is valid, i.e. derive an appropriate modelling of the robots and the environment. Simpler, yet valid models may lead to faster planning.

Since safe motions are computed off-line, they cannot fit to all the situations the robot will encounter while in on-line use. Thus, we propose as a second contribution a fast replanning method which computes a new safe motion from a previously optimal one in a very short CPU time (less than 2s). The idea of fast replanning motions is not new. Nishiwaki, et al, replanned motions using the mixture of pre-designed patterns [12], Tak, et al, [13] designed a method that balances dynamical motions, Yamane and Nakamura [14] combined motions from a database of motions generated using kinematics only, then used ”dynamics filters” to correct robot physical consistency, and Kagami et al. [15] relied on a control loop process to adapt the motion. The above techniques derive results with no guarantee, and still rely on closed-loop control to ensure constraint satisfaction a posteriori. Nevertheless, one may combine our guaranteed discretization approach with the above replanning techniques: any underlying time-grid discretization used to deal with a nonlinear inequality constraint that must be satisfied over a given time or space interval, may be replaced by our guaranteed discretization.

Our new replanning method is safe. It uses an inner approximation, i.e., a subset of feasible motions computed off-line in the vicinity of an optimal motion. In a previous work [11], we showed how to compute this feasible subset as a box. In this paper, we improve our method and show how to obtain a larger inner approximation. To overcome unpredicted situations, our fast replanning process consists of picking up, within the inner subset, a new motion that fits the new environment. Since it is no longer necessary to solve the nonlinear inequality constraints, replanning can then be done very rapidly. In this paper, we apply our planning and fast replanning methods to the kicking motion. In the case of soccer robots [16], this is the most important motion since it allows the robot to make a goal. Usually, kicking motion is computed off-line [17], and thus does not take into account the robot’s current position or the direction of the goal. Nevertheless, these pre-computed kicking motions allow the robot to react quickly to the situation, even if the kicking sometimes leads to an inaccurate ball trajectory. We show how to make the kicking motion more accurate by using our off-line safe motion planning and fast replanning processes. Moreover, in order to demonstrate the effectiveness of our replanning method we plan and replan a variety of kicking motions that we evaluate experimentally using the HOAP-3 robot.

The paper is organized as follows. Section II reviews the motion planning problem. Section III presents the first contribution of the paper; namely, the adaptive hybrid safe motion planning method. Section IV introduces the second contribution of the paper; namely, our method for fast replanning of safe motions. We illustrate our approach with the replanning of kicking motions. We conclude the paper by underlining the advantages and drawbacks of our safe motion planning framework and emphasize the prospective developments for our method.

II. Motion planning

In this section we describe the full-body model used for the robot under study. We give the optimality criterion and the equality and inequality constraints that must be satisfied by the sought-after motion.

A. Modeling of motion constraints

We consider humanoid robots as arborescent chains, with n degrees of freedom (dof). Since we focus on the lower part of the HOAP-3 humanoid robot, assuming the upper part fixed, we plan the trajectories for the $N_j = 12$ legs joints. We consider the motion safe if it ensures, during the whole motion duration, that joints position q_l, velocity $\dot{q}_l$ and torque Γ_l remain within acceptable bounds, i.e.

$$\begin{aligned}\forall l, \forall t \in [0,T] \quad & (\underline{q_l} \leq q_l(t) \leq \overline{q_l}) \\ & \wedge (\underline{\dot{q}_l} \leq \dot{q}_l(t) \leq \overline{\dot{q}_l}) \wedge (\underline{\Gamma_l} \leq \Gamma_l(t) \leq \overline{\Gamma_l}) \quad (1)\end{aligned}$$

that the robot does not slide, i.e.

$$\forall t \in [0,T] \quad F_X(t)^2 + F_Y(t)^2 \leq \mu^2 F_Z(t)^2, \quad (2)$$

where μ is the Coulomb friction parameter and F the contact force, and that the robot keeps balance, i.e. the motion keeps the ZMP [18] within the base of support, i.e.

$$\begin{aligned}\forall t \in [0,T] \quad & (\underline{ZMP_s} \leq ZMP_s(t) \leq \overline{ZMP_s}) \\ & \wedge (\underline{ZMP_f} \leq ZMP_f(t) \leq \overline{ZMP_f}). \quad (3)\end{aligned}$$

Eqs. (1)-(3) are then gathered in the following set of inequality constraints:

$$\forall i, \forall t \in [0,T] \quad g_i(q(t), \dot{q}(t), \ddot{q}(t)) \leq 0 \quad (4)$$

B. The motion planning problem

The motion planning problem is to find the set of optimal joint trajectories $\tilde{q}(t), \dot{\tilde{q}}(t), \ddot{\tilde{q}}(t)$ [to simplify notations, we assume that $\tilde{q}(t)$ also describe $\dot{\tilde{q}}(t)$ and $\ddot{\tilde{q}}(t)$.] that solves the problem:

$$\begin{array}{rl} \text{minimizes} & J(\tilde{q}(t)) \\ \text{subject to} & \forall i, \forall t \in [0,T] \quad g_i(\tilde{q}(t)) \leq 0 \\ \text{and} & \forall j, \forall \tau \in \{\tau_0, \ldots, \tau_k\} \quad h_j(\tilde{q}(\tau)) = 0 \end{array} \quad (5)$$

where J denotes the cost (or objective) function, g_i the set of inequality constraint functions, and h_j the set of equality constraint functions.

1) Cost function: The choice of the cost function $J(q(t))$ for motion planning must take into account the robot's features and the desired application. Some authors minimize motion duration [19] or jerk [3] for robot manipulators. In [2], the energy consumption taking into account actuators parameters (friction, etc.) is considered for humanoid robots. Biologically inspired cost functions can also be considered; for example, the minimum torque change [20]. In this paper we considered as criterion the motion duration in Sections II and III, and the energy consumption in Section IV.

2) Inequality constraint functions: The physical limits of the system are defined through the set of the inequality constraints $g_i(q(t))$ as shown previously. Hence, the robot's integrity and balance are ensured if these inequality constraints are satisfied over the whole motion duration, i.e., $\forall t \in [0, T]$.

3) Equality constraint functions: The set of the equality constraint functions $h_j(q(t))$ allows the definition of motion waypoints. These functions usually correspond to constraints on some of the system state variables at given time instants $\tau \in \{\tau_0, \ldots, \tau_k\}$, such as the beginning or the end of a motion.

4) The Semi-Infinite Programming problems and B-splines parametrization: Problem (5) is an optimal control problem, also called an Infinite Programming problem since it aims to find the continuous trajectories that satisfy a set of continuous inequality functions. Both the trajectories and the inequality functions can be decomposed into infinite sets of value. To the best of our knowledge, there are no algorithms able to deal with an Infinite Programming problem, so we have to transform it into a Semi-Infinite Programming (SIP) problem [21]. SIP is an optimization problem with a finite number of variables to optimize and a set of continuous constraint functions that is equivalent to an infinite number of discrete constraints to satisfy [22]. To do so, one usually uses a joint trajectory parametrization [23]:

$$q(t) = f(\mathbf{X}, t) \tag{6}$$

where $\mathbf{X}$ is a vector of parameters. We choose to compute joint trajectories with B-spline functions [24]. Thus, we define a motion via the parameter vector $\mathbf{X} = [T, p_{1,1}, p_{1,2}, \ldots, p_{N_j,N_s}]$ where N_s is the number of basis-functions, T is motion duration and $p_{k,j}$ the coefficients of the weighted sum of the B-spline functions. The joint trajectory $q_k(t)$ is computed as follows:

$$\forall k \in \{1, \ldots, N_j\} \quad q_k(t) = \sum_{j=1}^{N_s} p_{k,j} \times B_j(t) \tag{7}$$

Joint velocity and acceleration are obtained by differentiating (7). In this paper, we use nine uniform clamped B-splines basis functions. We gather the three first and the three last common basis-functions to obtain initial and final joint velocity and acceleration equal to zero and get the basis functions [24], hence we consider $N_s = 5$ optimization B-splines parameters per joint trajectory. Our motion planning problem boils down to finding a parameter vector $\tilde{\mathbf{X}}$ that is the solution of the following constrained optimization problem:

$$\begin{array}{rll} \text{minimize} & J(\mathbf{X}) & \\ \text{subject to} & \forall i, \forall t \in [0, T] & g_i(\mathbf{X}, t) \leq 0 \\ \text{and} & \forall j, \forall \tau \in \{\tau_0, \ldots, \tau_k\} & h_j(\mathbf{X}, \tau) = 0 \end{array} \tag{8}$$

Note that the inequality constraint must be satisfied over the whole time duration.

C. The classical method for solving SIP

Most classical constrained optimization algorithms, such as IPOPT [25] or FSQP [26] use a finite number of discrete constraints, hence require the discretization of continuous functions [22], [27]. Discretization usually consists of picking up the functions values over several time points taken on a grid. This leads to the replacement of the inequality constraints in Equation (8) by:

$$\begin{array}{c} \forall i, \forall t_k \in \mathbb{T} \quad g_i(\mathbf{X}, t_k) \leq 0 \\ \text{where } \mathbb{T} = \{t_1, ..., t_{M-1}, t_M\} \end{array} \tag{9}$$

Consequently, the continuous set of inequality functions (8) $\forall i, \forall t \in [0, T] \quad g_i(\mathbf{X}, t)$ becomes a discrete one: $\forall i, \forall t_k \in \mathbb{T} \quad g_i(\mathbf{X}, t_k)$ where the constraints are only considered for discrete values taken on the time-grid $\mathbb{T}$. Some methods run several optimization processes and modify the grid $\mathbb{T}$ in order to get better results [22]. In fact, the optimal value depends on the number of time points considered [28].

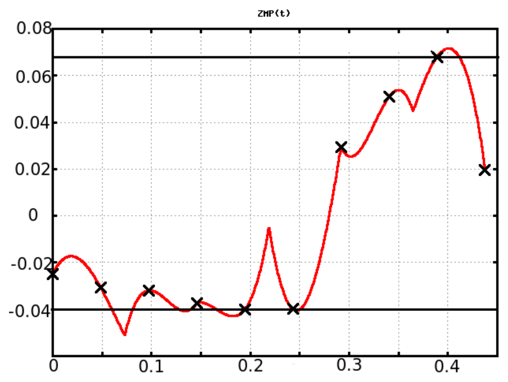


Fig. 1. Representation of a constraint function (the ZMP in the sagittal plane), obtained with a motion planning method using time-grid discretization.

We used this classical method of discretization with a sagittal 2D model of the HOAP-3 robot, and found out that it ensures constraint satisfaction only for the time instants taken on the time-grid (see Figure 1) [10], as was already mentioned in [29]. Furthermore, no information is given regarding constraint satisfaction between two points on the time-grid. Therefore, the constraints can be violated during the motion. To be able to compare our safe planning method and this classical way of discretization, we run the motion planning process several times using the classical discretization approach for several time-grid size. The outcomes are gathered on Table I. It highlights the fact that some time-grid sizes may produce hazardous motions. To find an appropriate time-grid size, i.e. that ensures the robot's safety, one usually performs several trials while increasing grid size, until one finds a satisfactory motion. Summing up the trial running times, a satisfactory motion is obtained in 22 CPU minutes (the checking process time excluded).

Remark 1: In the sequel, all CPU times were obtained on the following hardware and software. CPU : Intel Core 2 Duo E4400 2GHZ, Bus Speed: 800MHz, L2 Cache: 2MB, Memory: 2GB at 667MHz, OS: Linux Ubuntu 8.04. Constrained optimization problems were solved using IPOPT software package. Interval arithmetics related software was written in C++ and compiled using gcc-4.1

TABLE I
CRITERION (MOTION DURATION), COMPUTATION TIME, AND NUMBER OF VIOLATED CONSTRAINTS OBTAINED FOR DIFFERENT TIME-GRID SIZES.

grid size	criterion(s)	CPU time	violation
7	0.3691	49.9 s	17
13	0.3896	1mn 24s	14
31	0.3914	1mn 05s	14
61	0.3925	2mn 34s	6
121	0.3943	4mn 47s	2
301	0.4010	13mn 21 s	0
601	0.4169	24mn 9s	0

III. SAFE MOTION PLANNING

Before introducing our method for planning safe motions, we will introduce interval analysis and a guaranteed discretization approach, the two main ingredients of our technique.

A. Interval Analysis

Interval analysis was initially developed to account for the quantification errors introduced by the floating point representation of real numbers with computers, and it was then extended to validated numerics [30], [31]. A real interval $[a] = [\underline{a};\ \bar{a}]$ is a connected and closed subset of $\mathbb{R}$, with $\underline{a} = \mathrm{Inf}([a])$ and $\bar{a} = \mathrm{Sup}([a])$. The set of all real intervals of $\mathbb{R}$ is denoted by $\mathbb{IR}$. Real arithmetic operations are extended to intervals. Consider an operator $\circ \in \{+, -, *, \div\}$ and $[a]$ and $[b]$ two intervals. Then: $[a] \circ [b] = [\inf_{u \in [a], v \in [b]} u \circ v, \quad \sup_{u \in [a], v \in [b]} u \circ v]$.

Consider a function $m : \mathbb{R}^{n_1} \longmapsto \mathbb{R}^{n_2}$; the range of this function over an interval vector [a] is given by: $m([\mathbf{a}]) = \{m(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\}$. The interval function $[m] : \mathbb{IR}^{n_1} \longmapsto \mathbb{IR}^{n_2}$ is an inclusion function for m if $\forall [\mathbf{a}] \in \mathbb{IR}^n,\ m([\mathbf{a}]) \subseteq [m]([\mathbf{a}])$. An inclusion function of m can be obtained by replacing each occurrence of a real variable by the corresponding interval and each standard function by its interval counterpart. The resulting function is called the natural inclusion function. The performances of the inclusion function depend on the formal expression of m [30].

SIP problems have already been solved with constraint satisfaction and global optimization methods based on interval analysis ([31], and the references therein). These methods usually rely on branch-and-prune methods, whose complexity grows exponentially w.r.t. the dimension of the parameter vector, hence would require too long a computation time when used for motion planning with humanoid robots.

B. Guaranteed Discretization

The guaranteed discretization process ensures the validity of the inequality constraints over the whole motion duration [10], [11] by computing the minimum and the maximum values for the set of functions $g_i(\mathbf{X}, t)$ at a given interval $t \in [t]$. An *upper* bound for the maximum value $\max_{t \in [t]}(g_i(\mathbf{X}, t))$ is given by $\mathrm{Sup}([g_i](\mathbf{X}, [t]))$ and a *lower* bound for the minimum value $\min_{t \in [t]}(g_i(\mathbf{X}, t))$ is given by $\mathrm{Inf}([g_i](\mathbf{X}, [t]))$. Therefore, the upper bounds of $g_i(\mathbf{X}, t)$: max g_i are easily obtained by computing the upper bound of the inclusion function $[g_i](\mathbf{X}, [t])$ for a time interval $[t]$.

Using this guaranteed discretization approach, the inequality constraint functions in (8) are replaced by:

$$\forall i, \forall [t] \in \mathbb{IT} \qquad \mathrm{Sup}([g]_i(\mathbf{X}, [t])) \leq 0 \tag{10}$$

With $\mathbb{IT} = \{[t]_1, [t]_2], ..., [t]_{k-1}, [t]_k\}$ and $[t]_n = [t_{n-1}, t_n]$.

In practice, the bounds thus derived may be too coarse because of over-approximations in interval computation (the wrapping and dependence effects). Still, there are several techniques that can be used to obtain tighter enclosures by using, for instance, Taylor series expansion or some global optimization techniques [31]. In the sequel, we use a bisection process which decomposes an interval into 2^b subintervals to compute the minimal and maximal values of the constraint functions.

C. A Direct Method for Safe Motion Planning

The guaranteed discretization approach is used to plan a motion for the HOAP-3 robot using a sagittal 2D model [32]. The enclosures, which are a conservative computation of the extrema, are the values returned to the optimization algorithm. By doing so, the algorithm will be able to produce an optimal solution that satisfies all the constraints over whole motion duration.

We made several trials while increasing grid size. Table II shows that the criterion value reached with our safe method is lower, hence better that the criterion value reached by the classical method (Table I). It shows however that the CPU time required to obtain the result is clearly prohibitive. We will introduce a hybrid method for planning safe motions within a CPU time similar to the ones required by classical unsafe methods.

TABLE II
CRITERION, COMPUTATION TIME, AND NUMBER OF VIOLATED CONSTRAINTS AS OBTAINED FOR DIFFERENT CHOICES FOR THE TIME-INTERVAL VECTOR k AND THE BISECTION ORDER b.

k	b	criterion (s)	CPU time
6	15	0.3970	44h 03mn
12	15	0.3933	28h 36mn
6	16	0.3950	32h 19mn

D. A Hybrid Method for Safe Motion Planning

The main idea to reduce CPU time is to use guaranteed discretization as seldom as possible. Therefore, our idea is to develop an iterative process which uses classical discretization

Algorithm 1 Hybrid Safe Motion Planning

```
1:  $\mathbf{X} := \mathbf{X}_{init}$, $r := 1$, $\forall i, k\ \nu_{i,k,1} := 0$,
2:  repeat
3:      Use time-grid discretization and solve SIP with inequality constraint $\forall i, \forall t_k \in \mathbb{T} \quad g_i(\mathbf{X}, t_k) \leq -\nu_{i,k,r}$
4:      Check constraint satisfaction for the computed feasible motion: compute violation excess $\mu_{i,k,r}$
5:      if ($\exists i, k, r$ such as $\mu_{i,k,r} \neq 0$) then
6:          $\nu_{i,k,r+1} := \nu_{i,k,r} + \mu_{i,k,r}$
7:          $r := r + 1$
8:      end if
9:  until $\mu_{i,k,r} = 0$
10: return $\mathbf{X}$ which characterizes a safe motion.
```

processes to solve the SIP problem, i.e. an unsafe method, and then uses guaranteed discretization to check that the inequality constraints are satisfied.

For an iteration r, we modify the inequality constraints of Eq. (9) by:

$$\forall i, \forall t_k \in \mathbb{T} \quad g_i(\mathbf{X}, t_k) \leq -\nu_{i,k,r} \tag{11}$$

and solve the SIP problem with a classical discretization. Using the computed parameter vector $\mathbf{X}$, we compute the constraint violation magnitude $\mu_{i,k,r}$ for each constraint, using guaranteed discretization.

$$\mu_{i,k,r} = \max(0, \text{Sup}([g]_i(\mathbf{X}, [t_{k-1}, t_k])), \text{Sup}([g]_i(\mathbf{X}, [t_k, t_{k+1}]))) \tag{12}$$

If we detect a constraint violation ($\exists i, k, r$ such as $\mu_{i,k,r} \neq 0$), we penalize the constraint function over the corresponding time points, as follows:

$$\nu_{i,k,r+1} = \nu_{i,k,r} + \mu_{i,k,r} \tag{13}$$

and redo the optimization process again until no violation occurs. This algorithm is shown on Algorithm 1 and is summarized on Figures 2 and 3.

Figures 2 and 3 show the time history of the ZMP in the sagittal plane for the first two iterations. On Figure 2, a constraint violation is detected and the excess magnitude is computed; on Figure 3, a penalization is introduced, which further lowers the constraint limit. Eventually, no violation occurs and the constraint function remains within feasible values. For the ZMP constraint in the sagittal plane, only two iterations were needed, but other constraints needed more than two iterations. Table III shows the CPU time of the successive optimization processes. The total CPU time is $(41.4 + 33.8 + 7.6 + 8.4 + 4 \times 380 = 1611.2s)$, nearly 27 minutes.

TABLE III
CPU TIME OF THE HYBRID METHOD FOR MOTION PLANNING
(TIME OF THE CHECKING PROCESS IS 380S.)

r	CPU time (s)	maximal violation magnitude	total CPU time (s)
1	41.4s	5.4 %	421.4
2	33.8s	0.25 %	835.2
3	7.6s	$1e^{-5}$ %	1222.8
4	8.4s	no violation	1611.2

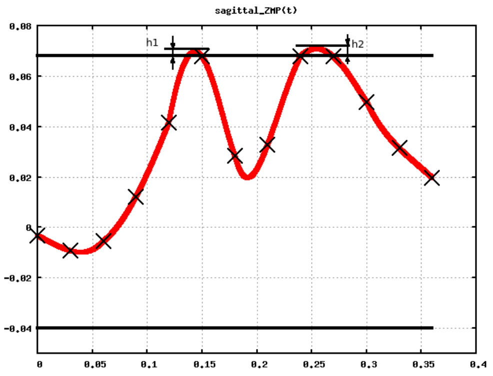


Fig. 2. Hybrid method for safe motion planning: first iteration $r = 1$

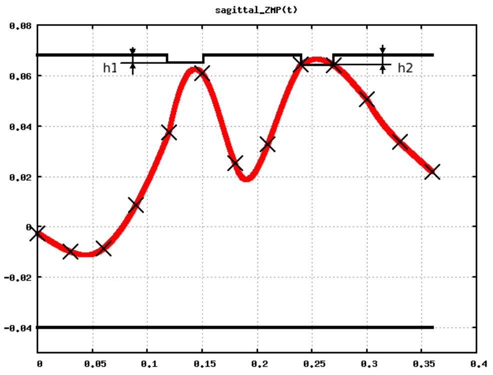


Fig. 3. Hybrid method for safe motion planning: second iteration $r = 2$

E. Convergence of the hybrid method

We will now analyze the convergence properties of the hybrid method and show that it converges to a conservative solution. Let us consider the time grid $\mathbb{T} = \{t_1, ..., t_{M-1}, t_M\}$. Denote $\mathbf{X}^\star \in \mathbb{R}^n$ the actual solution vector of (8), and let us define, for all i and k the scalar $\nu^\star_{i,k}$ as follows

$$\begin{array}{l} \text{if } (\exists\, \check{t} \in [t_{k-1},\, t_{k+1}],\ \text{such that } g_i(\mathbf{X}^\star, \check{t}) = 0) \\ \quad \text{then } \nu^\star_{i,k} = -g_i(\mathbf{X}^\star, t_k), \\ \quad \text{else } \nu^\star_{i,k} = 0. \end{array} \tag{14}$$

In fact, $\nu^\star_{i,k}$ is non null only when the inequality constraint $g_i(.,t)$ is active on $[t_{k-1},\, t_{k+1}]$. Then, it is easy to prove that $\mathbf{X}^\star$ is also a solution of

$$\left\{ \begin{array}{rl} \text{minimize} & J(\mathbf{X}) \\ \text{subject to} & \forall i, \forall k \in \mathbb{T} \quad g_i(\mathbf{X}, t_k) \leq -\nu^\star_{i,k} \\ \text{and} & \forall j, \forall \tau \in \{\tau_0, \ldots, \tau_k\} \quad h_j(\mathbf{X}, \tau) = 0. \end{array} \right. \tag{15}$$

(15) is a constrained optimization problem where the inequality constraints involve $\nu^\star_{i,k}$.

TABLE IV
COMPARISON OF THE CPU TIMES FOR THE METHODS PRESENTED.

methods	classical unsafe	direct safe	hybrid safe
CPU time	22mn	28h 36mn	27mn

Denote $\tilde{\mathbf{X}}_r$ the solution vector of

$$\left\{\begin{array}{ll} \text{minimize} & J(\mathbf{X}) \\ \text{subject to} & \forall i, \forall k \in \mathbb{T} \quad g_i(\mathbf{X}, t_k) \le -\nu_{i,k,r} \\ \quad\text{and} & \forall j, \forall \tau \in \{\tau_0, \dots, \tau_k\} \quad h_j(\mathbf{X}, \tau) = 0 \end{array}\right. \tag{16}$$

where the inequality constraints now involve $\nu_{i,k,r}$ computed via (13).

Proposition 1 (Convergence to a conservative solution): It exists $\tilde{r}$ such that

- $\forall r \ge \tilde{r}$, $\forall i$, $\forall k \quad \mu_{i,k,r} = 0$, i.e. the time grid discretization yields a safe motion,
- $\forall i$, $\forall k \quad \nu_{i,k,\tilde{r}} = \sum_{r=0}^{r=\tilde{r}} \mu_{i,k,r} \ge \nu^\star_{i,k}$.
- $\tilde{\mathbf{X}}_{\tilde{r}}$ is a conservative solution of (8), i.e. $\tilde{\mathbf{X}}_{\tilde{r}}$ is a feasible solution of (8) and $F(\tilde{\mathbf{X}}_{\tilde{r}}) \ge F(\mathbf{X}^\star)$.

Proof: From (13), $\forall r \ \nu_{i,k,r} \ge \nu_{i,k,r-1}$. Hence, it exists r_1 such that $\nu_{i,k,r_1} \ge \nu^\star_{i,k}$. Consequently, the feasible solution vector of (16) for $r = r_1$ is also a feasible solution vector of (15). Then $\forall i$, $\forall k$, $\forall r \ge \tilde{r} \quad \mu_{i,k,r} = 0$. ■

F. Computation time

Table IV shows the CPU times required by the classical unsafe method, the direct safe method and the hybrid safe method for motion planning. The classical method uses a time-grid discretization and produces a result in 22 minutes, but can provoke constraint violations that may be hazardous for robot integrity and balance. The direct safe motion planning method uses a guaranteed discretization of time, but requires prohibitive CPU time. Our hybrid method produces safe motions while requiring only 27 minutes. It ensures the robot's safety at the price of a CPU time only 20% longer than an unsafe method, which we consider acceptable.

G. Experimental validation

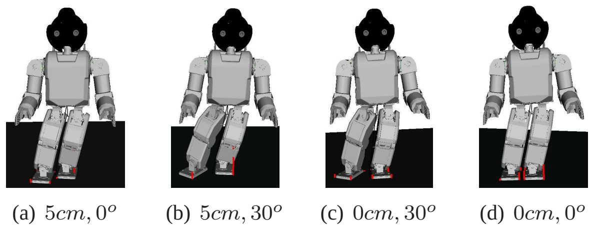


Fig. 4. Set of the four posture when the robot leans on its left foot, define by the parameters (L: feet distance, α: step direction).

To validate our hybrid motion planning method, we create a database of motions, to allow the robot to track a moving target. First, we define a posture using three parameters (l : feet distance in the frontal plan, L : step size in the sagittal plan, α : step direction). We choose four postures, with $l = 2cm$ as presented in Figure 4. We define a step as a motion which allows switching from one posture to another. We establish a simple heuristic algorithm (cf. Figure 5) to choose the next step to track the position of the target. This algorithm computes the direction θ of the target and chooses the best step to keep the target in front of the robot. Figure 6 shows the results of the experiment with the HOAP-3 robot. For a better understanding, the full video is available at ieeexplore.ieee.org. It emphasizes that the hybrid motion planning method ensures the validity of the constraint, since the robot tracks the target and keeps its balance without any balance controller.

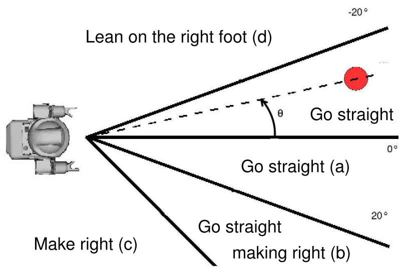


Fig. 5. Heuristic to choose the next step.

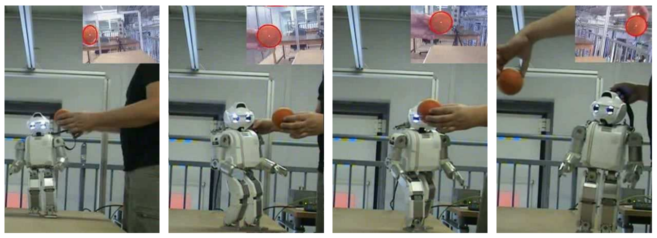


Fig. 6. Experimental validation of the motion computed with the hybrid motion planning method.

Remark 2: Unfortunately, we considered only 24 motions with initial and final static postures. By considering dynamic motion transition we might get smoother navigations, as done in [33].

H. Conclusion

In this section, we introduce an iterative hybrid method for planning safe motions, which requires CPU time of the same order of magnitude as for classical unsafe methods. For the HOAP-3 robot, we find that the required CPU time is only 20% longer, which is very acceptable. Then, by using a tracking experiment without any balance controller, we prove that our hybrid motion planning method is able to generate safe motions.

IV. Fast Replanning of Safe Motions

A. Replanning a kicking motion

In the above sections, we showed how to plan safe motions. We will now address fast replanning of such safe motions. Let us use the hybrid method introduced in the previous section to plan a safe kicking motion. The objective is to kick a ball located at $x = 1$cm. The impact point height is taken at $h = 3$cm, as shown in Figure 7, and we assume that the impact occurs at mid-time. Hence, the planned motion is merely characterized by the position (x, h) of the foot at the mid-duration time instant. The planned safe motion is depicted on Figure 8(a): The impact occurs at the desired position ($x = 1$cm, $h = 3$cm). Now, what happens if the ball is not at the expected position or has a wrong size ? Figure 8(b) shows the kicking motion obtained when the motion planned for $x = 1$cm is run while the ball is at position $x = 3$cm. In fact, the foot hits the ball at an impact point which is higher than expected. As a consequence, the energy transmitted to the ball may be insufficient to reach the desired goal. Figure 9 shows the outcome when the kicking motion planned for ($x = 1$cm, $h = 3$cm) is used with a ball smaller than expected, hence would have required a lower impact point, and which is also located at a bad position, i.e. $x = 3$cm instead of $x = 1$cm. Here the robot's foot goes over the ball and the robot falls. For a better understanding, the full video is also available at ieeexplore.ieee.org.

To improve and adapt the kicking motion, one solution may be to solve the new constrained optimization problem to generate a new optimal motion with the new equality constraint corresponding to the actual ball location, i.e. $x = 3$cm, or size. However, such an approach is often time-consuming. One can use a control loop process to modify the motion [12], [13], [14], [15], but those methods are based on simple model (such as cart-table) and focus mainly on the balance of the robot without considering all the constraints, such as the joint position or torque limits. Such approaches are of course unsafe, hence risky for robot's integrity.

We will now introduce a method which modifies the previous optimal kicking motion in a safe way, i.e., while ensuring constraint satisfaction, but at the price of only a very brief CPU time, i.e., less than 2 seconds. Our idea consists of replacing the set of inequality constraints $\forall t \in [0,T],\ g(\mathbf{X},t) \leq 0$, which is inherent to a given robot, by a set of bounds on the parameter vector, i.e., by $\mathbf{X} \in [\mathbf{X}]$, where $[\mathbf{X}]$ is an inner approximation, i.e., a subset, of the feasible set of parameters. Here we consider only inner approximations that are given as axis-aligned boxes. In our approach, inequality constraints that can be nonlinear and time-consuming to evaluate are merely replaced by bounds on the parameter vector. By doing so, on-line adaptation, i.e., on-line replanning will consist of an optimization process with bounds only on the parameters, new equality constraints h'_k and possibly a cost function J':

$$\begin{array}{rl} \text{minimizes} & J'(\hat{\mathbf{X}}) \\ \text{subject to} & \hat{\mathbf{X}} \in [\mathbf{X}] \\ \text{and} & \forall k \quad h'_k(\hat{\mathbf{X}}) = 0 \end{array} \qquad (17)$$

where $h'_k(\hat{\mathbf{X}})$ is the new set of equality constraints that characterizes the actual position of the ball, i.e. $x = 3$cm, or the actual impact point. It now remains to compute the feasible subset $[\mathbf{X}]$.

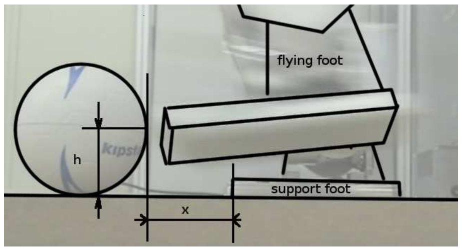
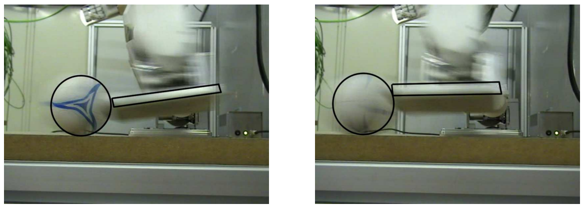


Fig. 7. Representation of the parameter (x, h) of a kicking motion.



(a) Ball at expected position (b) Ball at unexpected position

Fig. 8. A kicking motion planned for a ball at position $x = 1cm$, is used with a ball at expected and unexpected positions.

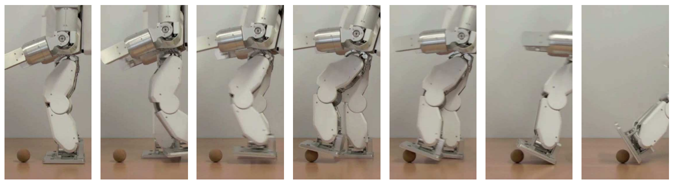


Fig. 9. A kicking motion planned for a ball at position $x = 1cm$ and impact point $h = 3cm$, is used with a smaller ball that would have required a lower impact point, and which is at an unexpected position. The robot falls.

B. Computation of the Feasible Subset

Let us denote $\tilde{\mathbf{X}}$ the parameter vector characterizing the optimal safe motion obtained using our hybrid method. To make the robot able to adapt its motion to as many situations as possible, we have to compute an inner approximation $[\mathbf{X}]$ of the feasible set, as large as possible, that contains the optimal vector $\tilde{\mathbf{X}}$ and satisfies all the inequality constraint functions. Recent studies addressed the computation of feasible sets using interval analysis for the design of parallel or serial robots (e.g., [34]). In fact, we do not need to compute the whole feasible set, but only an inner approximation of it. We will look for a subset $[\mathbf{X}]$ that will be contained in the feasible set. To obtain a box $[\mathbf{X}]$ as large as possible, we follow two steps: in the first step, we compute a feasible subset $[\mathbf{X}]$ that can be very small, and in the second step, we expand the feasible subset $[\mathbf{X}]$ to obtain a larger one.

1) First step: Computation of a feasible subset:

a) Principle: We start by computing the interval vector $[\mathbf{W}]$ as a weighted interval vector that will allow us to ignore or give emphasis to some components of the actual feasible set. In this paper, we propose to compute $[\mathbf{W}]$ by using the distance between the optimal vector $\tilde{\mathbf{X}}$ and the first constraint violation along each direction, as depicted on Figure 10. Hence, we can write a first guess for the feasible subset as:

$$[\mathbf{X}] = \tilde{\mathbf{X}} + [\mathbf{W}] \qquad (18)$$

Note that $0 \in [\mathbf{W}]$. Then, we prune the inconsistent parts of $[\mathbf{X}]$ by solving the following constrained optimization problem for scalar δ

$$\begin{array}{rl} \text{maximize} & \delta \in \mathbb{R} \\ \text{such that} & \forall j,\ \forall \mathbf{X} \in [\mathbf{X}],\ \forall t \in [0,T]\ g_j(\mathbf{X},t) \leq 0 \\ \text{where} & [\mathbf{X}] = \tilde{\mathbf{X}} + \delta \times [\mathbf{W}] \\ \text{and} & 0 \leq \delta \leq 1 \end{array} \qquad (19)$$

b) Algorithm: The principle of the algorithm is to start from $\delta = \delta_0 = 1$, and then to reduce it until the box $[\mathbf{X}] = \tilde{\mathbf{X}} + \delta_{final} \times [\mathbf{W}]$ no longer contains inconsistent vectors. Figure 10 shows the principle of our algorithm for computing the feasible subset $[\mathbf{X}]$. Using a branching algorithm with consistency tests as implemented in the ALIAS toolbox [35], we solve the following problem for box $[\mathbf{z}]$:

$$\begin{array}{rl} \text{find} & [\mathbf{z}] \subseteq [\mathbf{X}] \\ \text{where} & [\mathbf{X}] = \tilde{\mathbf{X}} + \delta_k[\mathbf{W}] \\ \text{such as } \exists j, \exists t \in [t] & \text{Sup}[g]_j([\mathbf{z}], t) > 0 \end{array} \tag{20}$$

where k is initially taken as $k = 0$. If the algorithm finds a solution, i.e., a box $[\mathbf{z}] \neq \emptyset$, it stops and δ_k is updated. New δ_{k+1} is chosen such that:

$$[\mathbf{z}] \ \cap \ (\tilde{\mathbf{X}} + \delta_{k+1}[\mathbf{W}]) = \emptyset \tag{21}$$

Problem (20) is solved again with new δ_{k+1} until it admits no solution. When the latter occurs, an inner approximation for the feasible set, i.e., a feasible subset, has been found. Eventually, the feasible subset is given by:

$$[\mathbf{X}] = \tilde{\mathbf{X}} + \delta_{final}[\mathbf{W}] \tag{22}$$

2) Second step: Expansion of the feasible subset:

a) Principle: On Figure 11, we can see that the subset $[\mathbf{X}]$ is not as large as possible, and it could be extended on X_1^+ or X_2^- directions. Thus, we focus on an expansion step to obtain an extended feasible subset $[\mathbf{X}]$.

b) Formulation: Let us denote by m, the index of $\mathbf{X}$ vector components. Let us introduce $\mathbf{E}_m^\circ$, $\circ \in \{-1, 1\}$, a vector of same dimension as $\mathbf{X}$ which contains null intervals except for the m^{th} component, which contains interval $[-1, 0]$ for $\circ \equiv -$ (negative direction) or $[0, 1]$ for $\circ \equiv +$ (positive direction). We can expand subset $[\mathbf{X}]$ in $\circ$ direction on the m^{th} component, by solving the following problem:

$$\begin{array}{rl} \text{maximize} & \rho_m \in \mathbb{R}^+ \\ \text{with} & [\mathbf{X}]' = [\mathbf{X}] + \rho_m \times \mathbf{E}_m^\circ \\ \text{and } \forall i, \ \forall \mathbf{X} \in [\mathbf{X}], \ \forall t \in [0, T] & g_i(\mathbf{X}, t) < 0 \end{array} \tag{23}$$

When ρ_m is found, the expanded inner subset is taken as $[\mathbf{X}] \leftarrow [\mathbf{X}]'$. The whole expansion process implies expanding all the parameters and directions. However, we have to choose the order of parameter expansion as it will impact the final result. Indeed, Figure 11 shows that if we extend $[\mathbf{X}]$ on X_1^+ direction, we will reduce the magnitude of following possible expansion to X_2^- direction (and vice-versa). Therefore we have to rank the parameters for expansion regarding:

- the sensitivity of the foot's location with respect to parameters. We have to emphasize the parameters that will bring the greater modification to the foot's position.
- the sensitivity of the constraint functions with respect to parameters. To get a feasible subset as large as possible, we have to emphasize the parameters for which constraint functions are less sensitive.

To sum up, we have to rank first the parameters that maximize the variation of the foot's position without leading to constraint violation.

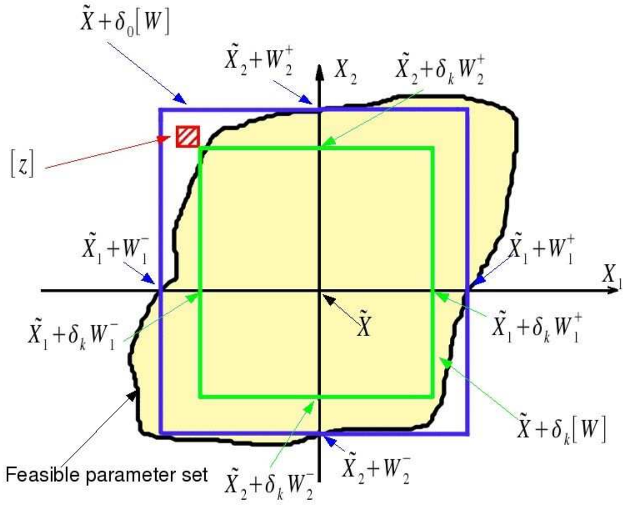


Fig. 10. Example of a feasible set and of its inner approximation : the feasible sub-set $[\mathbf{X}]$

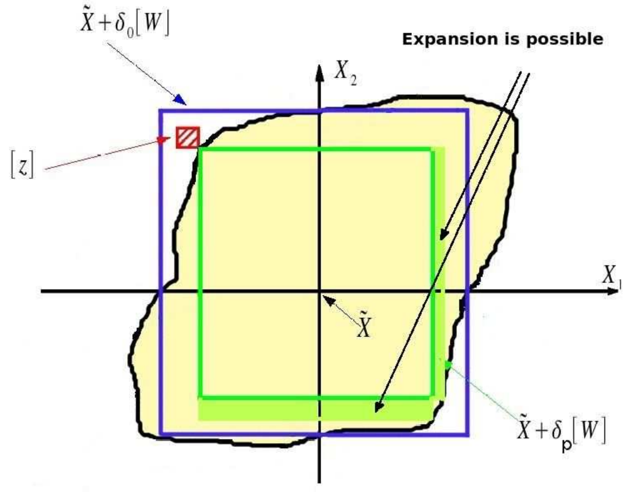


Fig. 11. Example of the expansion process. The final subset $[\mathbf{X}]$ rely on the order of the expansion.

C. Application to kicking motion

We apply now our fast safe replanning method to a kicking motion. Here, we detail the replanning results for the case ($x = 1$cm, $h = 3cm$).

1) Choice of the parameters to adapt: Obviously, it is not necessary to adapt all motion parameters. Since, we are interested in collision location along the x-axis, we adapt only the joint trajectories that impact motion in the sagittal plane: namely, the knees, hips pitch, and ankles pitch. Collision occurs at motion half-duration, thus we will only change the third B-spline parameters. Last, to proceed with the expansion, we rank the parameters according to the sensitivity criteria stated in the previous section. We choose to expand the feasible sub-set starting from the parameter of the flying leg, ankle, knee and hip, and then the supporting leg hip, knee and finally ankle.

2) Feasible Subset: Figure 12 depicts the results of our computation. We can see that the expansion process is effective and can indeed enlarge the feasible subset. The width of the feasible sub-set depends on the B-spline parameter under consideration, but it is interesting to see that the parameters can be changed within an interval of 7 to 14 degrees without making the robot fall, since our method ensures that no constraint is violated. Figure 13 shows the set of impact positions (x, h) that can be attained by the feasible set of motion parameters taken in $[\mathbf{X}]$. Let us assume we have initially planned a safe motion for $h = 3$cm and $x = 1$cm. Then, if we target an impact at $h = 3$cm, our replanning method can yield adapted kicking motions for balls located at $x \in [-1; 5]$cm.

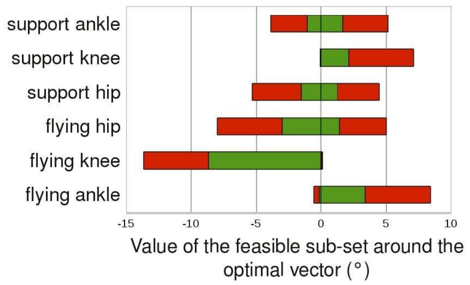


Fig. 12. Results of the computation of the feasible sub-set. The green values represent the feasible subset components prior to the expansion process. The red values represent the components after the expansion process.

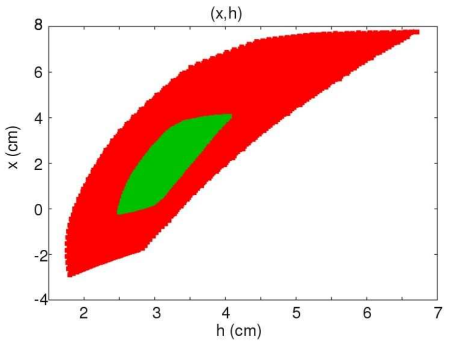


Fig. 13. Representation of the feasible set of impact position (x, h) for motions in the feasible subset $[\mathbf{X}]$. The green values are before the expansion process, the red values are after the expansion process.

D. Replanned motion

We choose to replan the optimal motion to make the robot kick a ball at $h = 3$cm high and located at position $x = 3$cm. Thus, we proceed with the optimization of the problem presented in Equation(17) but with the following equality constraints:

$$\begin{aligned} &\text{find } \hat{\mathbf{X}} \in [\mathbf{X}], \text{ such that} \\ &\quad h(\mathbf{X}, T/2) = 3cm \text{ and } x(\mathbf{X}, T/2) = 3cm \end{aligned} \tag{24}$$

Our implementation of the optimization algorithm took 1.5s of CPU time to find solution $\hat{\mathbf{X}}$. The replanned motion is depicted on Figure 14, with the impact at the desired position ($x = 3$cm, $h = 3$cm). Figures 15 depicts the time-history of the range of the feasible left foot velocity, the time-history of the planned and of the replanned left foot velocities, during the kicking motion. In fact, when replanning foot motion by modifying the third Bsplines parameters only, the velocities of the robot's joints remain unchanged at mid-duration, and joint values are only slightly modified. Consequently, the flying foot velocity remains almost the same for any replanned motion. Our fast replanning process for $x = 3$cm yields a kicking motion for the ball that is as good as the one originally planned for $x = 1$cm. In contrast, when the latter is used when the ball is at $x = 3$cm, ball motion has poorer performance.

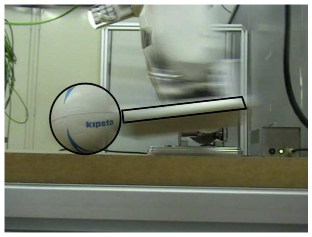


Fig. 14. Replanned kicking motion. See also full video at ieeexplore.ieee.org.

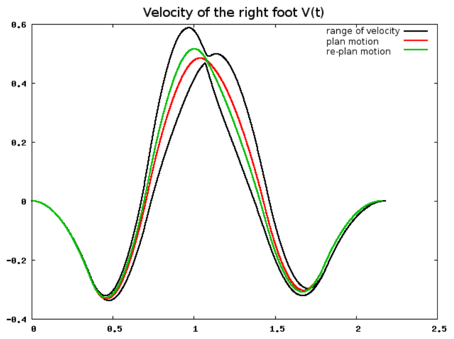


Fig. 15. Representation of the feasible time history of the velocity of the foot.

This emphasizes the effectiveness of our method, which is capable of producing new motions adapted to unpredicted environment, in a very brief CPU time.

E. Experimental evaluation

To demonstrate the effectiveness of our fast replanning method we investigate the replanning issue of a variety of kicking motions. Starting from the optimal motions planned for three different cases (distance to toe $x = 1$cm ; kicking impact point height h =1, 2 and 3 cm), we compute from these optimal solutions, the feasible sets of new impact positions that can be re-planned using the fast approach. The feasible subset in the vicinity of the optimal motion computed for the impact point ($x = 1$cm, $h = 3$cm) is already given in Figure 13, and the ones around the impact points ($x = 1$cm, $h = 1$cm) and ($x = 1$cm, $h = 2$cm) are gathered in Figure 16. Then, using these sets, we study the possibility to replan kicking motions using our fast method for twenty five new ball locations and impact heights, combining x =1, 2, 3, 4, 5 cm and h =1, 2, 3, 4, 5 cm, from the three originally planned motions. It must be noted also that this experimental evaluation of the method explores a large portion of the kicking motion's workspace for the HOAP-3 robot, i.e. the geometrical space reachable by the robot's flying foot. Fourteen out of the twenty five new impact points are included in at least one feasible subset, hence can be successfully replanned using our fast approach. Figure 17 shows the twenty five impact points (ball location $\times$ impact height), and the three feasible subsets used: an impact point is feasible if it is contained in at least one feasible subset. The maximum CPU time required for each fast replanning was less than 2 s. Figure 18 shows the replanned kicking motion for ($x = 2$cm, $h = 2$cm) as obtained from the optimal motion computed for ($x = 1$cm, $h = 3$cm): the robot now hits the ball, that is smaller than expected, in an appropriate manner while keeping balance. For a better understanding, the full video is also available at ieeexplore.ieee.org. Figure 19 shows the fast replanned motion for a kicking motion at ($x = 5$cm, $h = 3$cm) as obtained from an optimal kicking motion computed for ($x = 1$cm, $h = 1$cm). It is clear that our approach allows a large variation of the robot's foot motion while ensuring robot's balance and integrity. Here again, all experiments are done without any closed-loop balance controller. In summary, these experimental results are strong evidence that our method can be easily extended and applied to the fast replanning of arbitrary motions such as walking.

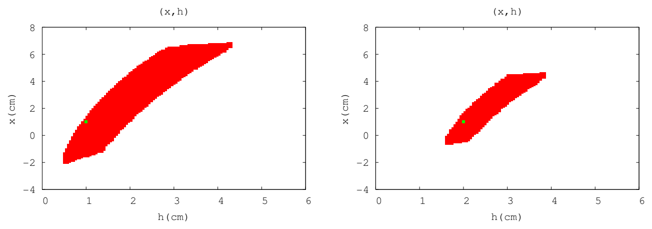


Fig. 16. Representation of the feasible sets as computed around the optimal motions ($x = 1$cm, $h = 1$cm) (left) and ($x = 1$cm, $h = 2$cm) (right).

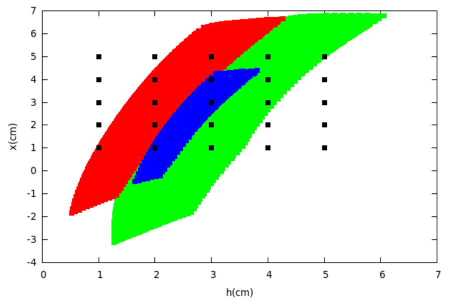


Fig. 17. Representation of the twenty five impact points (black dots) and the three feasible sets as computed around the optimal motions ($x = 1$cm, $h = 1$cm) (red color), ($x = 1$cm, $h = 2$cm) (blue color), and ($x = 1$cm, $h = 3$cm) (green color).

F. Conclusion

In the vicinity of each safe motion computed using our hybrid method, our fast replanning method computes a subset of feasible motions, i.e., motions that ensure the robot's safety and integrity. Thus, replanning boils down to solving an optimization problem with only bounds on parameters, and a set of new equality constraints. Nonlinear inequality constraints are no longer used, therefore replanning can be done very rapidly. We applied our approach to a variety of kicking motions and we were able to replan appropriate and safe kicking motions when the ball changed size and location in less than 2s CPU time. This is a very promising result since one can now consider performing replanning on-line and extend our replanning approach to arbitrary motions such as walking.

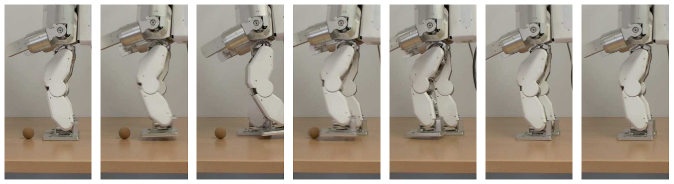


Fig. 18. Replanned kicking motion for ($x = 2$cm, $h = 2$cm) as obtained from the optimal motion computed for ($x = 1$cm, $h = 3$cm). See Fig. 9.

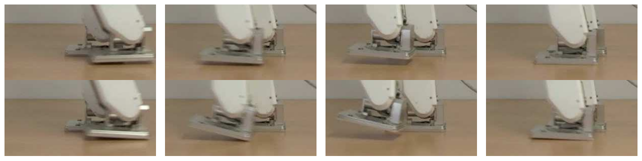


Fig. 19. Synchronized still images of the optimal motion for ($x = 1$cm, $h = 1$cm) (above) and the replanned one for ($x = 5$cm, $h = 3$cm) (below) as obtained from the former. Fast replanning process allows large variations of foot location.

V. Conclusion and Future Work

Planning motion for robots while ensuring their balance, integrity and safety requires that some constraints be satisfied for the whole motion duration, i.e., over a continuous interval of time. Classical motion planning approaches usually revert to time-grid discretization. We emphasize the fact that such approaches can be risky for robots since they cannot detect any constraint violation that may occur between two time-grid points. To address this shortcoming, we have introduced a hybrid method for planning safe motions. Our method is an iterative algorithm that combines a classical unsafe method with a verification step that checks constraint violation and computes excess via interval analysis. This method requires CPU times fairly similar to those of classical methods. To validate our approach, we built a database of such safe motions, then evaluated the database on a ball tracking experiment where the HOAP-3 robot successfully followed a moving ball without using any balance controller.

Safe motions are computed off-line and they are usually fitted only to a finite set of situations. If a robot meets an unexpected situation, a new motion must be replanned. The latter procedure used with thorough models is usually too time-consuming to be used on-line, even with unsafe methods. Therefore, we introduced a new method for replanning safe motions rapidly, i.e., in less than 2s CPU time.

We used our fast replanning method to replan a variety of safe kicking motions for a HOAP-3 robot. Starting from three safe kicking motions already available in the database, we were able to successfully replan new safe ones to kick balls of several sizes that were at a fairly large distance from the originally planned locations. Experimental evaluations gave strong evidence that our replanning approach can be extended to arbitrary movements such as walking.

The planning and fast replanning methods proposed in this paper are available as a C++ toolbox (on website safemotions. sourceforge.net). They were validated while planning motions for the lower body part of an HOAP-3 robot, using a 12-dof model.

If an impact model is available, our safe planning methods may easily be extended and used for planning safe motions with impact. It suffices to consider three phases: the two continuous motions before and after the impact, and the discrete event at the impact instant where the start of the motion after the impact may be obtained from an impact model and the end of the motion prior to the impact, as suggested in [36]. Furthermore, our safe methods can also be used with simpler, yet valid, models; they may yield results faster and eventually be used on-line.

In the near future, we will use and evaluate these methods for planning full-body motions, using models with a larger number of dof. Future work will address the presence of uncertainty in the considered models. One way to address such an issue is to implement an n-dimensional interval discretization within our hybrid method. We will have to use the guaranteed discretization approach on any uncertain variable, in addition to motion duration. Guaranteed Taylor series may also be investigated to reduce computation time. Our replanning method computes feasible inner subsets as axis-aligned boxes. We plan to develop an automatic way to rank the parameters during the expansion stage. For instance, sensitivity functions of the balance constraint function may be used, and one may expand first those parameters which have less impact on balance constraint. We will also investigate the possibility of using an alternative representation of sets, using ellipsoids or zonotopes for instance, to increase the size of the computed subsets, hence improving adaptation capabilities.

Acknowledgment

The authors would like to thank the Editor and the anonymous reviewers for their comments and suggestions which helped improving paper quality.

References

[1] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond, “Humanoid motion planning for dynamic tasks,” in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, Dec. 2005, pp. 1–6.

[2] 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, pp. 299–304.

[3] A. Piazzi and A. Visioli, “Global minimum-jerk trajectory planning of robot manipulators,” in *IEEE Transactions on Industrial Electronics*, vol. 47, febru 2000, pp. 140–149.

[4] W. Suleiman, E. Yoshida, J.-P. Laumond, and A. Monink, “On humanoid motion optimization,” in *IEEE-RAS 7th International Conference on Humanoid Robots*, 2007.

[5] O. Stasse, P. Evrard, N. Perrin, N. Mansard, and A. Kheddar, “Fast foot prints re-planning and motion generation during walking in physical human-humanoid interaction,” in *IEEE/RAS International Conference on Humanoid Robotics(Humanoids 09)*, 2009, pp. 284–289.

[6] P. Evrard, N. Mansard, O. Stasse, A. Kheddar, T. Schauss, C. Weber, A. Peer, and M. Buss, “Intercontinental, multimodal, wide-range telecooperation using a humanoid robot,” in *IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS)*, 2009, pp. 5635–5640.

[7] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, “Biped walking pattern generation by using preview control of zero-moment point,” in *IEEE International Conference on Robotics and Automation*, vol. 2, september 2003, pp. 1620 – 1626.

[8] ——, “Resolved momentum control: Humanoid motion planning based on the linear and angular momentum,” in *Intl. Conference on Intelligent Robots and Systems*, 2003.

[9] S. Lee and A. Goswami, “Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, april 2007, pp. 4667–4672.

[10] S. Lengagne, N. Ramdani, and P. Fraisse, “Safe motion planning computation for databasing balanced movement of humanoid robots,” in *EEE International Conference on Robotics and Automation, ICRA*, 2009, p. 1669–1674.

[11] ——, “Planning and fast re-planning of safe motions for humanoid robots : Application to a kicking motion,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 441–446.

[12] K. Nishiwaki, T. Sugihara, S. Kagami, M. Inaba, and H. Inoue, “Online mixture and connection of basic motions for humanoid walking control by footprint specification,” in *IEEE International Conference on Robotics and Automation, ICRA*, 2001, pp. 4110–4115.

[13] S. Tak, O.-Y. Song, and H.-S. Ko, “Motion balance filtering,” in *Eurocgraphics*, M. Gross and F. Hopgood, Eds., vol. 19, no. 3, 2000.

[14] K. Yamane and Y. Nakamura, “Dynamics filter - concept and implementation of online motion generator for human figures,” *Robotics and Automation, IEEE Transactions on*, vol. 19, pp. 421–432, 2003.

[15] S. Kagami, F. Kaneihiro, Y. Tamiya, M. Inaba, and H. Inoue, “Autobalancer: An online dynamic balance compensation scheme for humanoid robots, robotics,” in *The Algorithmic Perspective, Workshop on Algorithmic Foundations of Robotics*, 2001, pp. 329–340.

[16] S. Behnke, M. Schreiber, J. Stuckler, R. Renner, and H. Strasdat, “See, walk, and kick: Humanoid robots start to play soccer,” in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, Genova,, Dec. 2006, pp. 497–503.

[17] S. Carpin and E. Pagello, “The challenge of motion planning for human robots playing soccer,” in *Workshop on Humanoid Soccer Robots of the 2006 IEEE-RAS International Conference on Humanoid Robots*, December 2006, pp. 71–77.

[18] M. Vukobratovic and D. Juricic, “Contribution to the synthesis of biped gait,” *IEEE trans. Bio-Med Eng.*, vol. BME-16, pp. 1–6, 1969.

[19] A. Piazzi and A. Visioli, “Global minimum-time trajectory planning of mechanical manipulators using interval analysis,” *International Journal of Control*, vol. 71, pp. 631–652(22), November 1998.

[20] Y. Uno, M. Kawato, and R. Suzuki, “Formation and control of optimal trajectory in human multijoint arm movement,” *Biological Cybernetics*, vol. 6, no. 2, pp. 89–101, juin 1989.

[21] R. Reemtsen and J.-J. Rückmann, *Nonconvex optimization optimization and its applications : Semi-infinite Programming*, R. Reemtsen and J.-J. Rückmann, Eds. Kluwer Academic Publishers, 1998.

[22] R. Hettich and K. O. Kortanek, “Semi-infinite programming: theory, methods, and applications,” *SIAM Rev.*, vol. 35, no. 3, pp. 380–429, 1993.

[23] S.-H. Lee, J. Kim, F. Park, M. Kim, and J. E. Bobrow, “Newton-type algorithms for dynamics-based robot movement optimization,” in *IEEE Transactions on robotics*, vol. 21, 2005, pp. 657– 667.

[24] C. de Boor, *A Pratical Guide to Splines*. New York: Springer-Verlag, 1978, vol. 27.

[25] *Introduction to IPOPT : a tutorial for downloading, installing and using IPOPT*, april 7th 2006.

[26] 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*, Electrical Engineering Department, Institute for Systems Research University of Maryland, College Park, MD 20742.

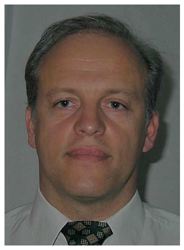
[27] O. von Stryk, “Numerical solution of optimal control problems by direct collocation,” 1993.

[28] R. Reemtsen, “Semi-infinite programming: discretization methods,” 1998.

[29] O. von Stryk and R. Bulirsch, “Direct and indirect methods for trajectory optimization,” *Ann. Oper. Res.*, vol. 37, no. 1-4, pp. 357–373, 1992.
[30] R. Moore, *Interval Analysis*. Englewood Cliffs: Prentice-Hall, 1966.
[31] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied interval analysis*, Springer, Ed. Springer, 2001.
[32] S. Lengagne, N. Ramdani, and P. Fraisse, “Guaranteed computation of constraints for safe path planning,” in *IEEE-RAS 7th International Conference on Humanoid Robots*, 2007, p. 312–317.
[33] J. J. J. Kuffner, K. Nishiwaki., S. Kagami., M. Inaba, and H. Inoue, “Footstep planning among obstacles for biped robots,” in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, Maui, HI, USA, 2001, pp. 500–505.
[34] N. Ramdani, M. Gouttefarde, F. Pierrot, and J. P. Merlet, “First results on the design of high speed parallel robots in presence of uncertainty,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept. 2008, pp. 2410–2415.
[35] “http://www-sop.inria.fr/coprin/logiciels/alias/.”
[36] T. Tsujita, A. Konno, S. Komizunai, Y. Nomura, T. Owa, T. Myojin, Y. Ayaz, and M. Uchiyama, “Humanoid robot motion generation for nailing task,” in *Advanced Intelligent Mechatronics, 2008. IEEE/ASME International Conference on*, Xian, July 2008, pp. 1024–1029.

Sébastien Lengagne received the Ph.D. degree in Computer Science from the University of Montpellier 2, France in 2009. His Ph.D. research was conducted at the Laboratoire d’Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), on planning and re-planning of safe motions for humanoid robots. Currently, he has a post-doctoral position in the Joint Robotics Laboratory (JRL), Tsukuba, Japan. His current research interests include the generation of multi-contact optimal dynamic motions for humanoid robots and avatars.

Nacim Ramdani received the Engineer degree from Ecole Centrale de Paris, France, in 1990, the Ph.D. degree from the University of Paris-Est Créteil, France, in 1994 and the Habilitation in 2005. Since september 2010, he has been a Professor at the Université of Orléans (IUT de Bourges) and member of the Laboratoire PRISME. From 1996 to 2010, he was Maître de Conférences with the University of Paris-Est Créteil. He was affiliated with the Robotics Department of LIRMM CNRS University of Montpellier 2 during 2005-2010 and also on secondment with the French National Research Institute in Computer Science and Control (INRIA) during 2007-2009. He is the chair of the action group on Interval Methods within the IEEE CSS TC on Computational Aspects of Control System Design, the co-chair of the workgroup on Set Computation Techniques within the French research group on Automatic Control (GDR MACS) and member of the IFAC TC on Discrete Events and Hybrid System. His current research interests include modeling and analysis of cyber-physical and autonomous systems in presence of uncertainty, and set membership estimation with applications to robotics, rehabilitation and human-robot interaction.



Philippe Fraisse received M.Sc degree in Electrical Engineering from Ecole Normale Superieure de Cachan in 1988. He received Ph.D. degree in Automatic Control in 1994. He is currently Professor at the University of Montpellier 2, France. He is the head of robotics department (LIRMM) and co-chair of French National Workgroup (GDR Robotique) working on Humanoid Robotics (GT7). He is also member of JRL-France scientific board (Japanese-French joint Laboratory for Robotics, AIST-JRL) and IEEE member. His research interests include modeling and control applied to robotic and rehabilitation fields, including humanoid robotics and robotics for rehabilitation.