HAL
archives-ouvertes.fr

A New Method for Generating Safe Motions for Humanoid Robots

Sebastien Lengagne, Nacim Ramdani, Philippe Fraisse

▶ **To cite this version:**

Sebastien Lengagne, Nacim Ramdani, Philippe Fraisse. A New Method for Generating Safe Motions for Humanoid Robots. Humanoids, Dec 2008, Daejeon, South Korea. 8th IEEE-RAS International Conference on Humanoid Robots, 2008. <lirmm-00431299>

HAL Id: lirmm-00431299

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00431299

Submitted on 12 Nov 2009

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.

A new method for generating safe motions for humanoid robots

Sébastien Lengagne [1,2], Nacim Ramdani [1,3], Philippe Fraisse [1,4]

[1] *LIRMM UMR 5506 CNRS Montpellier-France : www.lirmm.fr*

[2] *project team DEMAR, INRIA Sophia Antipolis Méditerranée : www−sop.inria.fr/demar*

[3] *project team COPRIN, INRIA Sophia Antipolis Méditerranée : www−sop.inria.fr/coprin*

[4] *University of Montpellier 2 : www.univ−montp2.fr*

Mail : lengagne,ramdani,fraisse@lirmm.fr

***Abstract*—This paper introduces a new method for planning safe motions for complex systems such as humanoid robots. Motion planning consists on finding the best joint trajectories. By using trajectory parameterization, the motion planning problem can be seen as a Semi-Infinite Programming problem (SIP) since it involves a finite number of parameters over an infinite set of constraints. Most methods solve the SIP problem by transforming it into a finite programming one by using a discretization over a prescribed grid. We show that this approach is risky because it can lead to motions which violate one or several constraints. Then we introduce our new method for planning safe motions. It uses Interval Analysis techniques in order to achieve a safe discretization of the constraints. We show how to implement this method and use it with state-of-the-art constrained optimization packages. Then, we illustrate its capabilities for planning safe motions for the HOAP-3 humanoid robot.**

***Index Terms*—Motion planning, Semi Infinite Programming, Discretization, Interval Analysis, Constraints.**

I. Introduction

Motion planning is a classical topic in robotics research. For complex systems such as humanoid robots, motion planning deals with a great number of constraints, which increases the computation time. Therefore, motion planning is usually done off-line to create a database of benchmark motions [1]. Then a simple on-line control loop can track one of these motions.

Motion planning includes the problem of digital actors' locomotion [2], kick motion generation on HRP-2 robot [3], computing a manipulator robot's trajectory [4] or smoothing pre-calculated motions [5]. In these works, the planned motions minimize a cost function and validate sets of equality and inequality constraints. With a joint trajectory parameterization, we transform the motion planning problem into a Semi-Infinite Programming problem (SIP), since it involves a finite number of parameters over an infinite continuous set of constraints.

To solve a SIP problem, the set of continuous inequality constraints are usually discretized by picking up several values over a given grid. Therefore, the obtained motions will satisfy the constraints only for the grid nodes. However, between two nodes the retained motion may violate some constraints that can have disastrous consequences on the integrity of the systems.

This paper presents a new method for planning safe motions, i.e. motions which ensure that the inequality constraints remain satisfied all over the motion duration. Our method uses the same optimization algorithms as classical one but replace the time-point discretization by a safe discretization that computes the constraints over time-intervals using Interval Analysis [6]. Interval Analysis has already been used in order to solve in a guaranteed way the problems of self-collision avoidance and prevention for the arms of a 2-degrees of freedom robot [7] or to solve the problem of finding collision-free paths [8].

A preliminary version of our safe discretization method was tested successfully on a two degrees of freedom pendulum where an optimal one-step motion was generated [9]. In the present paper, we address motion planning issues for a more complex system with six degrees of freedom. To be able to yield results in tractable CPU time, our method has been significantly improved. To illustrate the capabilities of our new method, we generate optimal step motions for the HOAP-3 humanoid robot in the sagital plane while considering two different cost functions and also using two optimization algorithms: C-FSQP [10] and IPOPT [11].

This paper is organized as follows: in section II we introduce the 2-D model of the humanoid robot used to generate motion in the sagital plane. Section III presents how the motion planning problem is transformed into a Semi-Infinite Programming problem, by a joint trajectory parameterization and emphasizes how risky the usual way of constraint discretization is. Section IV introduces the main ideas of our safe motion planning method and its practical implementation via Interval Analysis. In section V, the comparison between the classical and the safe motion planning methods is exposed for the two analyzed cost functions.

II. Modeling

A. 6 dof serial chain

Let us consider the motions of the HOAP-3 humanoid robot in the sagital plane and suppose that the balance in the frontal plane may be controlled using, for instance, the method presented in [12]. As a consequence, we can reduce the 3-D humanoid robot into a 2-D plane system, assuming that the upper parts of the body are equivalent to the chest (Cf. Fig 1). Furthermore, we will consider a serial chain starting from

the fixed foot and ending at the flying foot with 6 degrees of freedom (2 ankles, 2 knees, 2 hips).

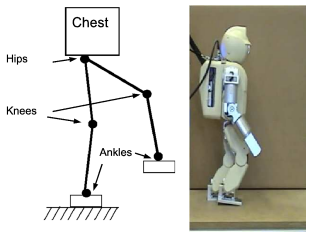


Fig. 1. 2-D model of the humanoid robot for a one-step motion in the sagital plane

B. Joint trajectory

We define a motion via the vector $\mathbf{X} = [\tau, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_6]$ where τ is the motion duration and $\mathbf{p}_i$ the coefficients of the weighted sum of B-spline functions which model the i^{th} joint position trajectory $q_i(t)$, as follows:

$$q_i(t) = \sum_{j=0}^{N_s} p_{i,j} \times B_j(t) \tag{1}$$

The joint velocity and acceleration are obtained by differentiating 1.

$$\dot{q}_i(t) = \sum_{j=0}^{N_s} p_{i,j} \times \dot{B}_j(t) \tag{2}$$

$$\ddot{q}_i(t) = \sum_{j=0}^{N_s} p_{i,j} \times \ddot{B}_j(t) \tag{3}$$

C. Dynamic modeling

The dynamic model equation allows to compute the joint torques $\Gamma(t)$ knowing the joint angle, velocity and acceleration vectors. This paper deals with the optimization of motions in the sagital plane, therefore we use the classical Newton-Euler method [13] for computing the joint torques of 2-D robots:

$$\Gamma(t) = NE(q(t), \dot{q}(t), \ddot{q}(t), t) \tag{4}$$

D. Balance

The computation of the Zero Moment Point (ZMP) gives information about the balance of humanoid robots. The ZMP is defined in [14] as a point, on the contact surface, where total inertia force is equal to zero. If this point stays within the base of support, the robot will keep its balance. The ZMP depends on the joint angle, velocity and acceleration.

$$ZMP(t) = f(q(t), \dot{q}(t), \ddot{q}(t), t). \tag{5}$$

III. Motion planning

A. Semi-Infinite Programming

For motion planning, one usually solves a constrained optimization problem in order to find out the optimal joint trajectory $q(t)$ which:

$$\begin{aligned} \text{minimizes} \quad & F(q(t)) && (6)\\ \text{subject to} \quad & \forall i, \forall t \in [0,T] \quad g_i(q(t)) \leq 0 && (7)\\ \text{and} \quad & \forall j \quad h_j(q(t)) = 0 && (8) \end{aligned}$$

where F denotes the cost or objective function, g_i the set of inequality constraint functions and h_i the set of equality constraint functions.

This is a classical optimal control problem where there are a continuous function: $q(t)$ and a set of continuous inequality constraints $g_i(q(t))$. Therefore we have to deal with an infinite dimensional problem. The B-spline parameterization allow to have a finite number of value to optimized and transform the problem into finding out the optimal parameter vector $\mathbf{X}$ which:

$$\begin{aligned} \text{minimizes} \quad & F(\mathbf{X}) && (9)\\ \text{subject to} \quad & \forall i, \forall t \in [0,T] \quad g_i(\mathbf{X}, t) \leq 0 && (10)\\ \text{and} \quad & \forall j \quad h_j(\mathbf{X}) = 0 && (11) \end{aligned}$$

This problem is defined as a Semi infinite Programming problem (SIP) , because there is a finite number of constraints which must be satisfied over a continuous time that can be expressed as an infinite number of instant t [15].

B. Cost function

The choice of the cost function $F(\mathbf{X})$ for motion planning must take into account the features of the robot and the desired application. Some authors minimize motion duration [16] or jerk [4] for robot manipulators. Some others minimize the energy consumption taking into account the parameters of the motors (friction, ...) [3] or a biological inspired cost function: the torque change [17].

In this paper we investigate two different cost functions:

- the motion duration $F(\mathbf{X}) = \int_0^T 1 dt$ (Cf. Sections III-F and IV-D)
- the mechanical energy consumption which is the integral of the Euclidean norm of joint torques $F(\mathbf{X}) = \int_0^T \Gamma^T \Gamma dt$ (Cf. Section V)

C. Equality constraint functions

The set of the equality constraint functions $h_j(\mathbf{X})$ allows to define the motion. These functions usually correspond to constraints on some system state variables at given time instants such as the beginning or the end of a motion. In our case, we consider six equality constraints which are the position of the flying foot at the beginning and at the end of the motion.

D. Inequality constraint functions

The set of the inequality constraints $g_i(\mathbf{X})$ is build from the physical limits of the system. These inequalities must be satisfied over the whole motion duration. In these experiments we consider limitations on the joint position, velocity and torque values and also on the ZMP location in order to ensure

the robot balance. The set of inequality constraint functions is as follows:

$$\begin{cases} \underline{q} \leq q(t) \leq \overline{q} \\ \underline{\dot{q}} \leq \dot{q}(t) \leq \overline{\dot{q}} \\ \underline{\Gamma} \leq \Gamma(t) \leq \overline{\Gamma} \\ \underline{ZMP} \leq ZMP(t) \leq \overline{ZMP} \end{cases} \quad (12)$$

E. Classical constraint discretization

Classical optimization algorithms use a finite number of discrete constraints. Hence constraints must be discretized. We present the classical way of discretizing them and we emphasize the fact that some constraints can be violated.

In the context of motion planning, discretization usually consists in picking up several time points in a grid for computing the constraint functions [18], [15]. Therefore, the inequality constraints in (10) are replaced by:

$$\forall i, \forall t_k \in \mathbf{T} \qquad g_i(\mathbf{X}, t_k) \leq 0 \quad (13)$$

where $\mathbf{T} = \{t_0 = 0, t_1, ..., t_{N-1}, t_N = T\}$.

Consequently, the continuous problem (10) where appears $\forall t \in [0, T]$ becomes a discrete one: $\forall t_k \in \{0, t_1, ..., t_{N-1}, T\}$ where the constraints are only satisfied for discrete values over the time-grid. Therefore, the optimization algorithm deals with a set of discrete values corresponding to the continuous constraint values over a time-grid. There are several methods to adapt the grid $\mathbf{T}$ in order to get better results [15].

F. The constraints are violated

Using the classical discretization method makes the optimization produces a solution which satisfies the constraints only for the discrete instants $\{t_0, t_1 ..., t_{N-1}, t_N\}$, but does not ensure them elsewhere. In order to highlight how such an approach may be hazardous for the robot integrity and balance, we used this method to plan a one-step motion for the HOAP-3 humanoid robot. We considered the constraint (12) and used the motion duration as the objective function and solve this optimization problem thanks to the algorithm C-FSQP [10].

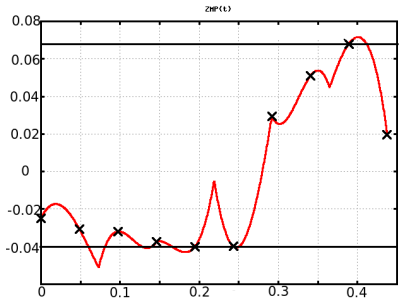
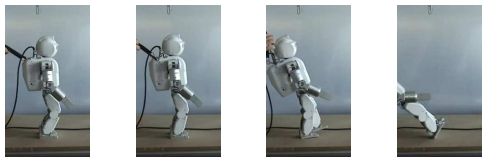


Fig. 2. Representation of the ZMP, for a motion optimized with a 10-time-points discretization

Figure 2 shows the ZMP time-history for an optimal motion obtained with a 10-time-point grid discretization. Here, in order to keep the balance of the robot, the ZMP must stay within the interval $[-0,04; 0,068]$ which corresponds to the robot foot size. This constraint is indeed satisfied for the discrete time instants, actually given to the optimization algorithm. However the continuous function is violated, for instance between $t = 0.05$s and $t = 0.1$s. To exhibit how this violation may impact the planned motion, we experimented this motion on the HOAP-3 humanoid robot and found that the robot lost its balance (Cf. Fig.3).



(a) $t = 0s$ (b) $t = 0.12s$ (c) $t = 0.24s$ (d) $t = 0.36s$

Fig. 3. A 10-time-points discretization time-minimum motion optimization using C-FSQP [10]

One intuitive way to solve this issue could be to increase the number k of discrete time instants. But this will not ensure the avoidance of any constraint violation [9]. In the next section we show how our safe method can solve this issue.

IV. Safe motion planning

A. Definition

Safe motion planning uses the same optimization softwares as usual motion planning. However, the constraints are computed using the guaranteed discretization introduced in our previous work [9]. The main idea at the core of the guaranteed discretization method is to return the extremum values of the inequality constraint functions $g_i(\mathbf{X}, t)$ during a time interval $[t] = [\underline{t}, \overline{t}]$ to the optimization algorithm instead of the values taken over the time grid. Then, equation (10) is replaced by:

$$\forall i, \forall [t] \in \mathbf{IT} \qquad \max_{\forall \tau \in [t]} g_i(\mathbf{X}, \tau) \leq 0 \quad (14)$$

With $\mathbf{IT} = \{[t]_1, [t]_2], ..., [t]_{k-1}, [t]_k\}$ where time intervals $[t]_n = [t_{n-1}, t_n]$. Interval analysis will allow to do the practical implementation of this method.

B. 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 was extended to validated numerics [19], [6], [20]. A real interval $[a] = [\underline{a}; \bar{a}]$ is a connected and closed subset of $\mathbb{R}$. With $\underline{a} = Inf([a])$ and $\bar{a} = Sup([a])$. 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:

$$[a] \circ [b] = [inf_{u \in [a], v \in [b]} u \circ v, \quad sup_{u \in [a], v \in [b]} u \circ v] \quad (15)$$

Consider $\mathbf{m} : \mathbb{R}^n \longmapsto \mathbb{R}^m$; the range of this function over an interval vector [a] is given by:

$$\mathbf{m}([\mathbf{a}]) = \{\mathbf{m}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \quad (16)$$

The interval function $[\mathbf{m}] : \mathbb{IR}^n \longmapsto \mathbb{IR}^m$ is an inclusion function for $\mathbf{m}$ if

$$\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \mathbf{m}([\mathbf{a}]) \subseteq [\mathbf{m}]([\mathbf{a}]) \tag{17}$$

An inclusion function of $\mathbf{m}$ 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 $\mathbf{m}$.

Finally, interval analysis offers a practical but guaranteed mean to compute minimum and maximum values for a function $m(t)$ when t is defined over a given interval $[t]$. An *upper* bound for the maximum value $\max_{t\in[t]}(m(t))$ is given by $Sup[m]([t])$ and a *lower* bound for the minimum value $\min_{t\in[t]}(m(t))$ is given by $Inf[m]([t])$.

In practice, the bounds thus derived may be too large because of the wrapping and dependence effects. Still, there are several techniques that can be used to obtain tighter ones by using for instance Taylor series expansion or some global optimization techniques [21].

Therefore the upper bounds for (14) *max* g_i are obtained in an easy and practical way by computing the upper bound of the inclusion function $[g_i]$ for a time interval $[t]$:

$$\forall i, \forall [t] \in \mathbf{IT} \qquad Sup[g]_i(\mathbf{X}, [t]) \leq 0 \tag{18}$$

C. Practical application

In this study, we have used a simple way to solve the wrapping and dependence effects issue. In order to compute tighter enclosures for the inclusion functions, and hence a better evaluation of the extremum values of a given function over the time-interval, we have chosen to split the time-intervals.

Splitting is the process that decomposes an interval into N_b subintervals, then the extremum of the interval may be taken as the extremum of the N_b both sub-intervals. Figure 4 shows how the splitting makes the enclosure tighter. However, splitting may increase computation time. These splitting steps are carried out until the estimated maximum and minimum values are obtained with an acceptable accuracy.

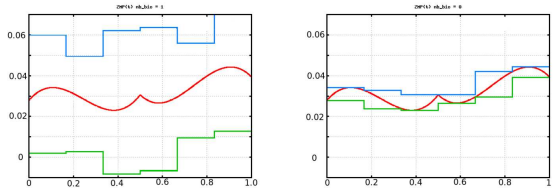
As an illustrative example, let us consider the inclusion function $[ZMP]([t])$ (Equation 5). In the latter equation, the t variable appears several times. Therefore the formal expression of the function $ZMP(t)$ will suffer from the dependence effect. Thanks to the splitting, we can control the accuracy in the computation of the extremum values.

Let us quantify the accuracy of the enclosures as the relative measure of width of the *computed* enclosures with regard to the *actual* one, as follows

$$eps(\%) = \frac{\text{width(computed)} - \text{width(actual)}}{\text{width(actual)}} \times 100 \tag{19}$$

Here, the actual enclosures are the ones obtained with $N_b = 2^{15}$ subintervals.

Table I shows that the bigger the of sub-intervals (N_b), the better the accuracy and the larger the computation time



(a) no bisection (b) 9 bisections

Fig. 4. The bounded ZMP over 6 time-intervals

T_c. Therefore the user has to make a compromise between accuracy and computation time. In our case we choose to consider 128 sub-intervals.

N_b	1	2	4	8	16	32	64	128	256
eps(%)	73	49	19	8.2	3.7	1.7	0.8	0.34	0.17
T_c(ms)	0.7	1.4	2.5	5.4	11	20	42	80	160

TABLE I
SPLITTING ACCURACY / COMPUTATION TIME

D. The constraints are satisfied

We experimented a 7cm step motion generated with our safe motion planning method considering 10 intervals, for the humanoid robot HOAP-3. Figure 6 shows this motion, where the robot keeps its balance contrary to figure 3 where it falls. Figure 5 shows both the ZMP time-history $ZMP(t)$ and the guaranteed interval discretization $[ZMP]([t]_k)$ always remain within the base of support.

$$\forall t \in [0, \tau] \quad -0,04 \leq ZMP(t) \leq 0,068 \tag{20}$$

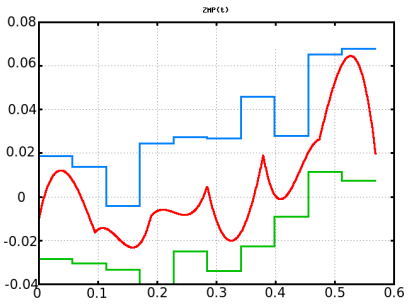
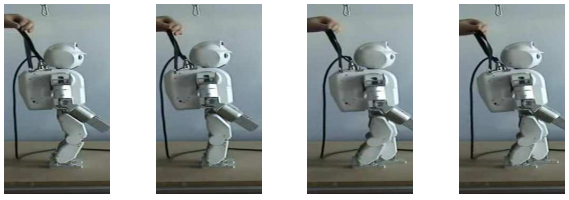


Fig. 5. Representation of the ZMP, for a motion optimized with 10 intervals



(a) $t = 0s$ (b) $t = 0.2s$ (c) $t = 0.4s$ (d) $t = 0.6s$

Fig. 6. A safe discretization time-minimum motion optimization using C-FSQP [10]

E. The gradient of constraint functions

Some optimization algorithms need the computation of the gradient of the constraint functions with respect to the parameter vector $\frac{\partial g(\mathbf{X},t)}{\partial \mathbf{X}}$. With a grid discretization (Cf Section III-E) this gradient is computed at the grid instant.

Now, the method used in subsection IV-C for solving equation (18) subdivides a given time interval $[t]_k$ into N_b subintervals $[t]_k = \cup_{l=1,\ldots,N_b}[t]_{k,l}$ Then there exists a subinterval $[t]_{k,l_{max}}$ which contains the maximum of $g_i(\mathbf{X},t)$ for $t \in [t]_k$. Since we do not keep track on when the maximum occurs within $[t]_{k,l_{max}}$, we choose to approximate the gradient of the maximum of $g_i(\mathbf{X},t)$ by the value of the gradient obtained at the middle of the subinterval $[t]_{k,l_{max}}$. This is summarized in the following equation:

$$\frac{\partial}{\partial \mathbf{X}} Sup[g]_i(\mathbf{X},[t]_k) \approx \frac{\partial}{\partial \mathbf{X}} g_i(\mathbf{X}, Mid[t]_{k,l_{max}}) \qquad (21)$$

We can infer from this that the number of sub-intervals N_b will influence the accuracy of the computed gradient and hence the efficiency of the (C-FSQP or IPOPT) optimization algorithm used. Note also that the partial derivatives can be computed either via formal methods or automatic differentiation [22]. In our work, we used the latter.

V. Results

A. Procedure

In this section we compare the usual motion planning method with our safe motion planning method. We want to plan a walking motion with steps of 5 cm. Then we consider a cyclic motion where the initial posture is symmetrical to the final one. We consider the 6 dof model of the HOAP-3 humanoid robot (Subsection. II-A). The joint trajectories are computed with 4 B-splines functions. The vector $\mathbf{X}$ contains $(4-1)\times 6+1 = 19$ parameters (Here -1 is due to the cyclic nature of the retained motion, i.e. final position is symmetric to the initial one) . We opt for the algorithm IPOPT [11] and perform two optimizations, the first one for minimizing the motion duration and the second one for minimizing the energy consumption. On one hand we asses the classical motion planning method considering a grid of 15 points. On the other hand we use our safe motion planning method considering 5 intervals splitted into 128 sub-intervals.

We repeated 500 times the numerical procedure with random initial conditions to evaluate the impact of our method regarding the repeatability and the time computation of the algorithm.

B. Repeatability

We define the repeatability of a cost function value the percentage of optimization procedure which produce a solution with this value. A high percentage of repeatability means that the optimization algorithm has good convergence properties. Tables II and III show that there are some differences of the cost function values between the classical and the safe motion planning method ($1.055 \neq 1.075$). These differences are mainly due to the fact that the classical motion planning method produces a solution which may violate some constraints, whereas a slower motion will not.

Moreover, one can see that the percentage of a failure of the optimization algorithm is higher with the classical motion planning method (4%) than with the safe motion planning method (0.4%). In the same time the percentage to have one of the two best cost function values is higher considering a safe motion planning (98.6 %) than the classical one (76.8 %). We can see the same phenomena on Tables IV and V, even if the difference is smaller (97.6% against 95.4%). This is due to the fact that an energy-minimum motion is slower than a time-minimum motion and consequently violates less the dynamic constraint functions. Therefore it appears that a guaranteed discretization ensures a better convergence for the optimization algorithm.

Cost function value	Repeatability (%)
1.055	52.4
1.125	24
1.135	17.2
1.215	0.2
1.225	0.2
1.64	2
failure	4

TABLE II
REPEATABILITY FOR A TIME-MINIMUM CLASSICAL MOTION PLANNING

Cost function value	Repeatability(%)
1.075	51,2
1.145	47.4
1.655	0.6
1.835	0.2
1.905	0.2
failure	0.4

TABLE III
REPEATABILITY FOR A TIME-MINIMUM SAFE MOTION PLANNING

Cost function value	Repeatability(%)
5.95	90.8
6.55	4.6
6.85	3.6
8.05	0.8
failure	0.2

TABLE IV
REPEATABILITY FOR AN ENERGY-MINIMUM CLASSICAL MOTION PLANNING

Cost function value	Repeatability(%)
5.95	89.2
6.55	8.4
6.85	1.6
8.15	0.06
failure	0.2

TABLE V
REPEATABILITY FOR AN ENERGY-MINIMUM SAFE MOTION PLANNING

C. Computation time

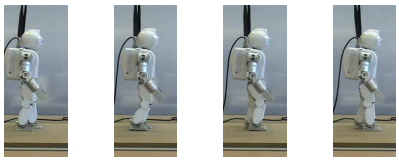
A study about the computation time of these two methods for a simple 2-dof system was done in [9]. Table VI shows the difference of the computation time between the classical and the safe motion planning method considering two different cost functions: the motion duration and the energy consumption. The computation time of the safe motion planning method is slower than the classical one, and the choice of the cost function has an effect on the computation time. Indeed a time-minimum motion has fast dynamics that cause constraint violation, thus the optimization algorithm will call more often the constraint functions than for a slower motion such as energy minimum motion. That is why safe motion planning method is 10 times slower than classical one for a time-minimum motion and only 4 times slower for energy-minimum motion.

motion planning method	classical	safe
time-minimum	8 s	90 s
energy-minimum	39 s	168 s

TABLE VI
COMPARISON OF THE COMPUTATION TIME

D. Experiments

Figure 7 shows an energy-minimum motion computed with the safe motion planning method. One can see that the safe motion planning method generates a motion which avoids any constraint violation since the robot keeps its balance.



(a) $t = 0s$ (b) $t = 0.8s$ (c) $t = 1.6s$ (d) $t = 2.4s$

Fig. 7. Energy-minimum 5cm step motion

VI. CONCLUSION

Motion planning consist on finding the best joint trajectory which minimizes a cost function and validate a set of equality and inequality constraints. Usually the joint trajectory is computed from a set of parameters. This parameterization allow to see the motion planning problem as a Semi-Infinite Programming (SIP) problem which is solved by transforming it into a finite one thanks to a time-grid discretization. Unfortunately the time-grid discretization can lead to some constraint violations which may impact the integrity and on the balance of the robot.

To the contrary our new safe motion planning method uses Interval Analysis to compute the maximum of the constraint functions over time-intervals, which avoids any constraint violation. We showed that our method increases the repeatability of the algorithm but the computation time is larger.

Here we addressed the one-dimension time discretization issue but the same approach can be used for N-dimensions space discretization. We illustrated our safe motion planning method with a 2-D model of humanoid robots. We plan to use it with a 3-D model.

REFERENCES

[1] J. Denk and G. Schmidt, "Synthesis of a Walking Primitive Database for a Humanoid Robot using Optimal Control Techniques," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, Tokyo, Japan, November 2001, pp. 319–326.

[2] 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.

[3] 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 *Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics*, 2006, pp. 299–304.

[4] 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.

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

[6] R. E. Moore and F. Bierbaum, *Methods and Applications of Interval Analysis (Siam Studies in Applied Mathematics, 2.)*. Soc for Industrial & Applied Math, 1979.

[7] H. Fang and J. P. Merlet, "Dynamic interference avoidance of 2-DOF robot arms using interval analysis," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Aug. 2005, pp. 3809–3814.

[8] L. Jaulin, "path planning using intervals and graphs," *Reliable Computing*, vol. 7, no. 1, pp. 1–15, fevrier 2001.

[9] 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.

[10] 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.

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

[12] P. Fraisse, S. Cotton, A. Murray, and F. Pierrot, "Towards dynamic balance control of humanoid robots by using com and zmp," in *IEEE-RAS 7th International Conference on Humanoid Robots*, 2007.

[13] W. Khalil and E. Dombre, *Modeling, Identification & Control of Robots*, 3rd ed., E. B. Heinemann, Ed. Hermes Sciences Europe, march 2002.

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

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

[16] 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), 10 November 1998.

[17] 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.

[18] O. von Stryk, "Numerical solution of optimal control problems by direct collocation," 1993.

[19] T. Sunaga, "Theory of interval algebra and its application to numerical analysis," *RAAG Memoirs, Ggujutsu Bunken Fukuy-kai*, vol. 2, pp. 547–564, 1958.

[20] A. Neumaier, *Interval methods for systems of equations*. Cambridge: Cambridge university press, 1990.

[21] E. Hansen and G. Walster, *Global optimization using interval analysis*, 2nd ed. Marcel Dekker, 2004.

[22] C. Bendtsen and O. Stauning, *FADBAD, a flexible c++ package for automatic differentation*.