Guaranteed Computation of Constraints for safe path Planning

Sebastien Lengagne, Nacim Ramdani, Philippe Fraisse

Guaranteed computation of constraints for safe path planning

Sébastien Lengagne, Nacim Ramdani[1] and Philippe Fraisse
DEMAR : LIRMM - UMR CNRS 5506 / Université Montpellier II / INRIA
161 rue Ada 34392 Montpellier Cedex 5 - France
{Sebastien.Lengagne,Nacim.Ramdani,Philippe.Fraisse}@lirmm.fr

Abstract— **Path planning issues are often solved via constrained optimization methods but with constraints which must be satisfied over a whole interval of time or space. The use of fast numerical toolboxes implementing state-of-the-art constrained needs to discretize the continous constraints over a time grid. Thus, the obtained solution, in this way, will satisfy the constraints only for time values corresponding to the time grid. Obviously, some constraints could be violated with catastrophic consequences when dealing with, for instance, the balance of humanoid robots. In this paper we introduce a guaranteed discretization method which uses interval analysis to ensure that the constraints are satisfied over the whole time interval. We analyze numerically this method by performing a trajectory generation under constraints dedicated to the motion of the HOAP-3 humanoid robot.**

INTRODUCTION

Path planning is an important topic in the humanoid robotics research field. Robot motion generation is usually achieved by solving a path planning problem which boils down to a constrained optimization one. This problem is solved by [1] for the planning of digital actors' locomotion ; by [2] for the optimization of motions such as a kick motion on HRP-2 robot or by [3] for the planning of a manipulator robot's trajectory.

The validity of the constraints during the whole motion is of prime importance. However, the optimization algorithms used need only the assessment of the constraints over a time or space grid. The *continuous* constraints must then be discretized to be taken into account in the optimization process. [2] and [4] use such a time-grid discretization, so they consider only discrete values of the continuous constraints. Therefore, these constraints are satisfied over the time-grid but can be violated elsewhere. This is the reason why we introduce a *time-intervals* dicretization method which guarantees the validity of the constraints for the whole motion.

This paper is structured as follows. First, we recall the path planning problem under constraints for generating optimal motions. Then, we present two simplified dynamic models considered to assess the proposed method : a double pendulum and the legs of the HOAP-3 humanoid robot expressed in the sagital plane. The next section will address the time discretization issue : we remind the classical discretization method usually used in robotics and then introduce our developments. In the last part, we compare the two discretization methods for the generation of a double pendulum motion. Finally, we apply the time-intervals method to the generation of a one-step motion with the HOAP-3 humanoid robot.

[1]Author Nacim Ramdani is on leave from CERTES EA 3481 Université Paris 12 Val de Marne

I. PATH PLANNING UNDER CONSTRAINTS

To generate a motion, the usual solution is to solve a constrained optimization problem which takes into account several constraints, and an objective function which depends on the application.

The constrained optimization problem is to find out the best parameters vector $\mathbf{x}$ which minimizes an objective function $F(\mathbf{x},t)$ and satisfies some constraints $g_j(\mathbf{x},t)$, $\forall j \in \{1,...,ng\}$. Therefore, the problem can be expressed as :

$$\begin{gathered} \min \int_0^T F(\mathbf{x},t)dt \\ \text{subject to} : \forall j, \forall t \in [0,T] \quad g_j(\mathbf{x},t) \leq 0 \end{gathered} \tag{1}$$

There can be some constraints on the vector parameters :

$$x_i^{min} \leq x_i \leq x_i^{max} \tag{2}$$

A. The constraints : $g_j(\boldsymbol{x},t)$

The constraint functions allow to take into account the limitations of the system such as the mechanical constraints (joint limits : angle, velocity, torque ...). They can also describe the desired behaviour. In this case the constraints are the position of some bodies of the robot at several time instants to define the motion. Specifically dedicated to humanoid robots, we add a function which describes the equilibrium of the robot (ZMP : Zero Moment Point).

We consider n_g constraint functions such as :

- the limit value for joint angle :

$$q_i^{min} \leq q_{i(t)} \leq q_i^{max} \tag{3}$$

- the limit value for joint velocity :

$$\dot{q}_i^{min} \leq \dot{q}_{i(t)} \leq \dot{q}_i^{max} \tag{4}$$

- the limit of the ZMP position (cf. II-A)

$$zmp_{min} \leq zmp_{(t)} \leq zmp_{max} \tag{5}$$

All these constraints can be expressed as :

$$\forall t \in [0,T] min_j \leq g_j(t) \leq max_j \quad \forall j \in \{1,...,n_g\} \tag{6}$$

To fit with the optimization problem (cf. 1), we need to modify the bounded contraints (eq:6) into two inequality constraints :

$$\begin{array}{ll} \forall t \in [0,T] & \\ -g_{j(t)} + min_j \leq 0 & \forall j \in \{1,...,n_g\} \\ g_{j(t)} - max_j \leq 0 & \end{array} \quad (7)$$

B. *The objective function : $F(\boldsymbol{x},t)$*

The choice of the objective function for motion optimization must take into account the features of the robot and the desired motion. For example, [2] defines the objective function as the electrical energy consumption while taking into account the parameter (friction, etc.) of the actuators; whereas [3] chooses to minimize the jerk.

This paper focuses on the computation of the constraints, so we will not deal with the objective function. The optimization parameters define the motion function. The next part describes how to compute the joint trajectories ($q_{(t)}$) from the vector **x**.

C. *Computation of joint values : $q_{(t)}$*

The vector **x** is comprised of :

- the motion duration t_{max},
- the initial joint value q_i,
- the final joint value q_f,
- a parameter vector **P** which modifies the trajectory between the initial and final joint values.

We have chosen to compute the joint value $q_{(t)}$ thanks to two functions which depend on the value of t_{max}, q_i and q_f :

- $s_{(t)}$ links q_i and q_f with initial and final velocity and acceleration equal to zero,
- $b_{(t)}$ are Bspline-functions.

1) the function $s_{(t)}$: .

$$s_{(t)} = at^5 + bt^4 + ct^3 + d \quad (8)$$

with :

$$\begin{aligned} a &= 6 \times (q_f - q_i)/t_{max}^5 \\ b &= -\tfrac{5}{2} \times a \times t_{max} \\ c &= \tfrac{5}{3} \times a \times t_{max}^2 \\ d &= q_i \end{aligned} \quad (9)$$

2) the functions $b_{(t)}$: $b_{(t)}$ is a set of n_b B-splines functions $b_{n(t)}$ as shown in figure:1.

3) joint value: The joint value $q_{i(t)}$ is computed by adding $s_{(t)}$ with a ponderation of $b_{j(t)}$. This ponderation is contained in the vector : $\mathbf{P}_{(n_j \times n_b)}$, where n_j is the number of joints and n_b the number of splines for each joint :

$$q_{i(t)} = s_{i(t)} + \sum_{n=1}^{n_b} \mathbf{P}_{(i,n)} b_{n(t)} \quad (10)$$

The joint velocity $\dot{q}_{i(t)}$ and acceleration $\ddot{q}_{i(t)}$ are computed by derivating $q_{i(t)}$.

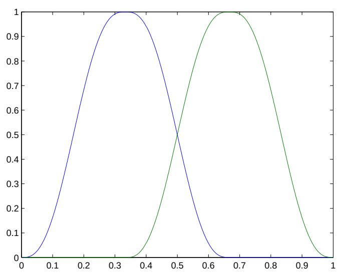


Fig. 1. Representation of B-splines functions $b_{n(t)}$ with $n_b = 2$.

So, we are in position to compute the joint angle ($q_{(t)}$), velocity ($\dot{q}_{(t)}$) and acceleration ($\ddot{q}_{(t)}$). These results are useful for the next section to calculate the constraints and objective functions.

II. Modeling

In this section, we present the dynamic model of a planar robot. Starting from the joint position, velocity and acceleration $(q(t), \dot{q}(t), \ddot{q}(t))$, we compute the torque $\Gamma(t)$ and ZMP value $zmp(t)$.

A. *Computation of the dynamic model*

The dynamic model equation allows to compute the joint torques $\Gamma_{(t)}$ knowing the joint angle, velocity and acceleration :

$$\Gamma_{(t)} = NE(q_{(t)}, \dot{q}_{(t)}, \ddot{q}_{(t)}, t) \quad (11)$$

In [5], the Newton-Euler method is used for computing the dynamic model of a 3-D robot thanks to a two-recursions-algorithm. This paper deals with optimization applied to 2-D robot, therefore we adapt the Newton-Euler method for computing the joint torques of a 2-D robots.

For humanoid robots, the computation of the Zero Moment point (ZMP) give information about the balance. [6] defines ZMP as the point zmp, on the contact surface, where the moment is equal to zero $M_{zmp} = 0$ (cf. fig 2). If this point stays in the base of support, the robot maintains its equilibrium.

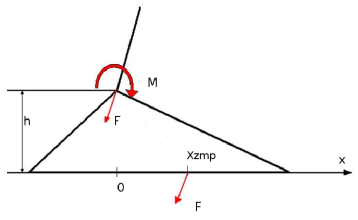


Fig. 2. Representation of zmp

The zmp location depends on the joint angle $q_{(t)}$, velocity $\dot{q}_{(t)}$ and acceleration $\ddot{q}_{(t)}$.

B. Double pendulum

The double pendulum (cf. fig.3) is used, in section IV, as a simple model. We define the initial value $q_i = [0,0]$ to get the foot position : $(x = 0,\ y = 0)$. The final joints value is computed to do a step lenght of d :

$$q_f = \left[Acos\left(\frac{d}{2L}\right), 2 \times Acos\left(\frac{d}{2L}\right) \right] \tag{12}$$

Let us consider one parameter per joint. So, in the case of Fig.3, we get 2 parameters : $\mathbf{P} = [p_1; p_2]$. The goal is to determine the best value of $\mathbf{P}$, which minimizes the energy consumption and guarantees the contraints shown in I-A, thanks to the constrained optimization algorithm.

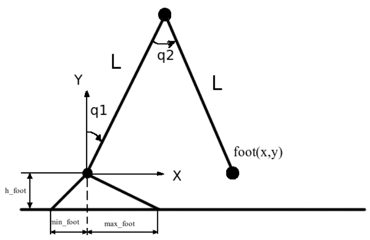


Fig. 3. Double pendulum used for the comparaison of the discretization method

We define the objective function as :

$$F(\mathbf{x}) = \sum_{t=0}^{t_f} \sum_{i} \Gamma^2_{i(t)} \tag{13}$$

Where $\Gamma_{i(t)}$ is the torque of joint i at the instant t:

C. HOAP-3 's leg in the sagittal plane

To validate the method proposed in section V, we will use a more complicated model : HOAP-3's legs. We consider only the lower limbs, so the upper parts of the body are equivalent to a mass on the chest (cf. fig:4). The legs of the humanoid robot HOAP-3 are modeled as a 6-links robot in the sagittal plane (hips, knees and ankles).

To apply the Newton Euler Algorithm (cf. II-A), we consider a fixed contact between one foot and the ground, whereas the other foot moves from its initial position to its final position.

D. Why discretization is needed ?

Starting from the parameter vector $\mathbf{x}$ we are able to compute the continuous functions : $(q(\mathbf{x},t), \dot{q}(\mathbf{x},t), \ddot{q}(\mathbf{x},t), \Gamma(\mathbf{x},t), zmp(\mathbf{x},t))$. Nevertheless, the constrained optimization software needs the evaluation of the constraints over a time grid t_i, $i = \{1,2,...,k\}$. Consequently, these constraints must be discretized. First we describe the method usually used in robotics for discretizing continuous functions (section : III-A), then we will introduce a new way of discretization which uses interval arithmetics (section : III-B).

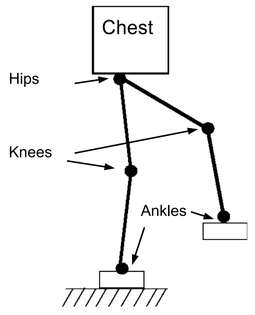


Fig. 4. 2-D model of HOAP-3

III. Guaranteed constraints computation

A. The classical time-grid discretization

Discretization addresses the process of transferring continuous models and equations into discrete counterparts. Usually discretization consists on picking up several time points of the functions to be discretized. The constraints are computed as shown : for $i = \{1,2,..,k\}$

$$min_j \leq g_j(t_i) \leq max_j \qquad \forall j \in \{1,...,n_g\} \tag{14}$$

Once the optimization is finished with optimal results, the motion satisfies all the constraints over the time grid but this does not ensure that the constraints are satisfied elsewhere (cf. Fig:5).

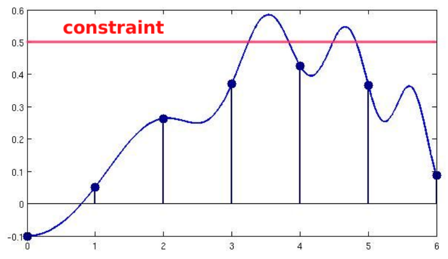


Fig. 5. Example of a function discretization. (The discrete values satisfy the constraint, whereas the continuous function violates it.)

Figure 5 shows the discretization of a constraint function. If the maximum value of the constraint is 0.5, the constrained optimization program does not detect any constraint violation even though the constraint is not satisfied for $t \in [3,4]$, and even though there is a computed value at $t = 3$ and at $t = 4$. In spite of time points discretization process, a solution which violates the continuous constraints may exist.

This drawback can be solved by increasing the number of time-points, but this will also increase the computation time [4]. Another solution may be to consider a time-interval instead of a time-point. This is the topic of the next part.

B. Time-interval discretization via interval analysis

The main idea of the time-interval discretization is to bound a function $g_j(t)$ with a minimum and maximum value during a time interval $[t] = [t_{min}, t_{max}]$ instead of computing a single value.

The issue is to compute $min_{t\in[t_{min},t_{max}]} g_j(t)$ and $max_{t\in[t_{min},t_{max}]} g_j(t)$. This operation is easily done thanks to 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 [7], [8], [9]. A real interval $[a] = [\underline{a}, \bar{a}]$ is a connected and closed subset of $\mathbb{R}$. 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:

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

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

$$\mathbf{g}([\mathbf{a}]) = \{\mathbf{g}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \qquad (16)$$

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

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

An inclusion function of $\mathbf{g}$ 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 this inclusion function depend on the formal expression for $\mathbf{g}$.

Interval analysis has been used in several fields. In robotics, it has been used to solve off-line the constrained optimization problem which furnishes the best trajectory for manipulator robots [3].

C. Computation of the constraints

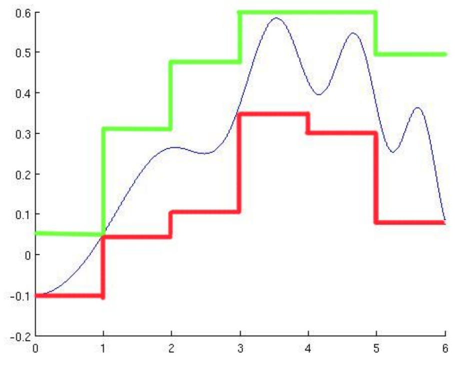


Fig. 6. Example of time-interval discretization of a function. (The minimum and maximum value are computed for each interval. The continuous value is always between those two values.)

Figure 6 shows the time-interval discretization of the function of figure 5. A drawback of interval arithmetics is the existence of pessimism (cf. [10]). The pessimism is the difference between the actual maximum, or minimum value of the function and the computed one. To deal with this problem, time intervals can be sub-divided into n_d subdivisions, and determine the minimum (or maximum) value for each subdivision. The minimum (or maximum) for the whole interval is the minimum (maximum) of all the minimum (maximum) of the subdivisions.

IV. Performance assessment

The programs are developped using C/C++ language. The constrained optimization is done thanks to FSQP program [11]. The interval arithmetic is done by PROFIL/BIAS [12].

We proposed to compare the two methods with a simple example : the double pendulum (cf. fig.II-B). We define the initial value $q_i = [0, 0]$ to get the foot position : $(x = 0,\ y = 0)$. The final joints value is computed to do a step of size d :

A. Number of contraints

We take into account the constraints :

- the joint position,velocity : 2 constraints,
- the zmp location : 1 constraint.

Starting from the interval $t = [0, t_{max}]$, we define k time instants and time-intervals, to be used in the constrained optimization. The algorithm considers only one inequality per computation.

$$n_c = 2k \times (2+1) \qquad (18)$$

B. Time-points discretization

Table I shows the results for different number (k) of discretization time points. n_e is the number of evaluation of the constraint value by the optimization program. p_1 and p_2 are the final value of the optimized parameters.

k	n_e	result	constraint	p_1	p_2
3	168	OK	no valid	-26	0.62
5	280	OK	no valid	-25.5	0.62
10	1203	ERROR	valid	-1.7	4.1
30	61037	Max iter	valid	-2.1	-2.3
100	19644	OK	no valid	-1.6	-3.35

TABLE I
RESULTS FOR DOUBLE PENDULUM OPTIMIZATION MOTION WITH INSTANT DISCRETIZATION.

The number (n_e) of constraints evaluations increases with the number (k) of time points. The final value (p_1, p_2) depends on the value of k. Therefore we cannot say which value (p_1, p_2) is the optimal one. Moreover, when the optimization program finishes with the optimal solution (result = OK), the constraints are not satisfied for the whole motion.

C. Interval discretization

Table II shows the results for different number (k) of intervals and different number of sub-division (n_d). The number n_e is also linked to the number of interval (k). Nevertheless the the final solution (p_1, p_2) is almost the same for several value of k.

An important result is that, with only four intervals (or subdivisions), the optimization will find the best solution ($p_1 = 0, p_2 = 0$) and the constraints are always satisfied.

k	n_d	n_e	result	constraint	p_1	p_2
1	1	/	ERROR	/	/	/
1	4	364	OK	valid	0	0
2	1	/	ERROR	/	/	/
2	2	298	OK	valid	0	0
2	4	607	OK	valid	0	0
4	1	414	OK	valid	0	0.05
4	4	942	OK	valid	0	0

TABLE II
RESULTS FOR DOUBLE PENDULUM OPTIMIZATION MOTION WITH INTERVAL DISCRETIZATION.

D. Assessment

This simple example shows that the time-point discretization carried out for motion optimization, can generate some solutions which violate the constraints. Whereas the time-interval discretization ensures the constraints validity. Moreover, the iteration number of the constraints computation is lower than for the time-instant discretisation ($942 << 19644$).

V. EXPERIMENTAL RESULTS

The guaranteed computation of the constraints is applied to the generation of joints trajectories for a one-step-motion of the humanoid robot : HOAP-3 (cf. II-C).

The optimization vector x is composed of :

- the initial and final value (q_i, q_f) for all joints : 2×6 parameters,
- one B-splines parameter for all joints : 1×6 parameters,
- the motion duration : 1 parameter.

Therefore the constrained optimization program deals with 19 parameters.

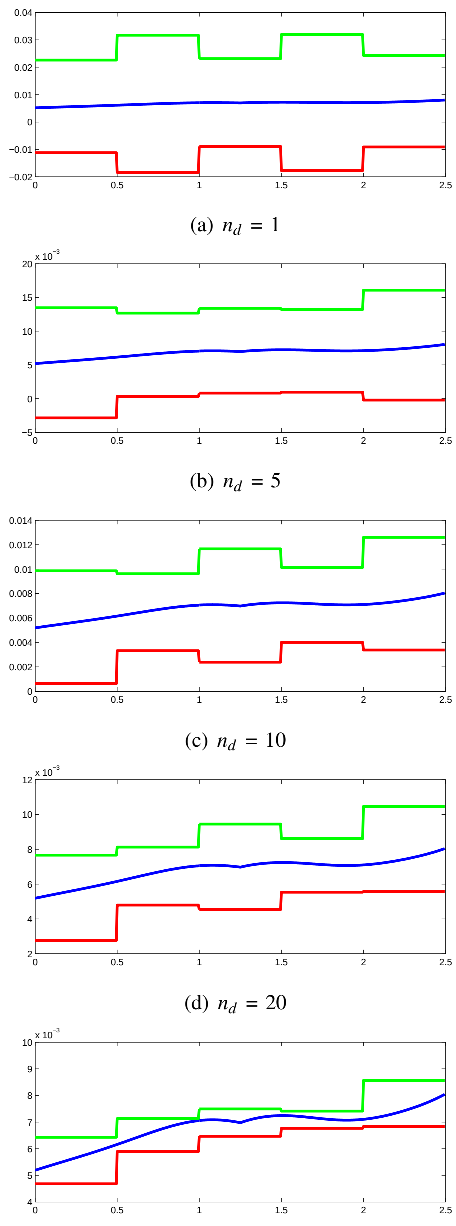
To avoid any convergence problem in this study, we do not take into account any objective function. We consider only the following constraints :

- the initial and final positions of the moving foot (position (x, y) of the toes and position (y) of the heel) : 2×3;
- the minimum and maximum limit of joint values and velocities for all the intervals: $2 \times 6 \times k$;
- the minimum and maximum values of the ZMP location : $zmp_{(t)}$, for all the intervals : $2 \times k$.

For this constrained optimization, we focus on 5 intervals of discretization : $k = 5$. Thus, we get $n_c = 6 + (12 + 2) \times k = 76$ constraints to validate. We also use $n_d = 5$ sub-divisions.

On figure 7, we compare the values of the $zmp_{(t)}$, computed from the obtained solution, and the value given by the time intervals computation for several number nd of sud-divisions.

When $n_d = 1$ (fig.7(a)), the minimum and maximum values of the function : $zmp_{(t)}$ are computed with an important pessimism and we know that the constrained optimization program will reject acceptable solutions. Whereas for $n_d = 100$ (fig.7(e)), the pessimism effect is very small. Nevertheless we know that the computation time increases with the value of n_d. A good compromise between large computation time and small pessimism effect is to choose a value of n_d between 5 and 20.


(a) n_d = 1

(b) n_d = 5

(c) n_d = 10

(d) n_d = 20

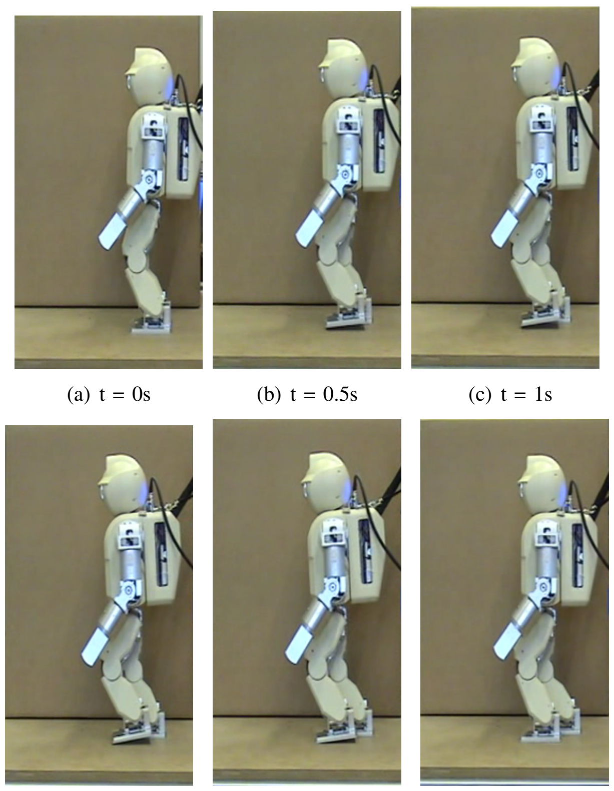
(e) n_d = 100

Fig. 7. Comparaison of interval computation of the ZMP for several n_d values

Figure 8 shows the experimentation of the retained motion, with the humanoid robot HOAP-3.

CONCLUSION

In this paper we have presented a new method that ensures the validity of continous constraints for trajectories optimization. We have shown that the commonly used time-point discretization method is fast but do not ensure the validity of the


(a) t = 0s (b) t = 0.5s (c) t = 1s

(d) t = 1.5s (e) t = 2s (f) t = 2.5s

Fig. 8. One step motion obtained with guaranteed discretization

constraints. Therefore the constrained optimization algorithm will accept some solutions which violate the contraints.

However, the time-interval method proprosed ensures constraints validity over the whole motion. We have compared the two methods for the contrained optimization of a double pendulum motion, and applied the time interval discretization to motion optimization of the humanoid robot HOAP-3.

We have planned to implement our method to address the contrained optimization of a whole step motion and to merge the two methods to decrease computation time.

References

[1] J. Pettré, J.-P. Laumond, and T. Siméon, "A 2-stages locomotion planner for digital actors," in *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 258–264.

[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 *Proceedings of the 2006 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] M. Hardt, K.Kreutz-Delgado, and J. Helton, "Optimal biped walking with a complete dynamical model," *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, pp. 2999–3004, 1999.

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

[6] M. Vukobratović and B. Borovac, "Zero-moment point : Thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 157–173, 2004.

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

[8] R. Moore, *Interval Analysis*. Englewood Cliffs: Prentice-Hall, 1966.

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

[10] L. Jaulin, M. Kieffer, K. Didrit, and E. Walter, *Applied interval analysis*, Springer, Ed. Springer, 2001.

[11] J. L. Z. Craig Lawrence 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 Constraints1*, Electrical Engineering Department and Institute for Systems Research University of Maryland, College Park, MD 20742.

[12] O. Knppel, *PROFIL/BIAS V2.0*, Technische Universitt Hamburg-Harburg Technische Iformatik II, D-21071 Hamburg Germany, february 1999.