Generation of Dynamic Motions Under Continuous Constraints: Efficient Computation Using B-Splines and Taylor polynomials

Sébastien Lengagne, Paul Mathieu, Abderrahmane Kheddar, Eiichi Yoshida

Generation of Dynamic Motions Under Continuous Constraints: Efficient Computation Using B-Splines and Taylor polynomials

Sébastien Lengagne, Paul Mathieu, Abderrahmane Kheddar and Eiichi Yoshida
CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/CRT, Tsukuba, Japan

***Abstract*— This paper proposes a new computation method to solve semi-infinite optimization problems for motion planning of robotic systems. Usually, this problem is solved by means of time-grid discretization of the continuous constraints. Unfortunately, discretization may lead to unsafe motions since there is no guarantee of constraint satisfaction between time samples. First, we show that constraints such as joint position and velocity do not need time-discretization to be checked. Then, we present the computation method based on Taylor polynomials to evaluate more complex constraints over time-intervals. This method also applies to continuous equality constraints, to continuous maximum derivative constraint, and to compute the cost function.**

***Index Terms*— Semi-Infinite Programming, motion optimization, Taylor polynomials, humanoid robots**

INTRODUCTION

Computing motions for complex robotic systems is still an open issue. Optimization techniques were early used to solve robotic motion planning; they proved to be efficient for local problems. Global motion planning turns to be more efficiently solved using probabilistic approaches. Recently, task-space based approaches, where trajectories are generated implicitly and in a reactive way, appear to be attractive alternative and can be efficiently combined with probabilistic planning to reduce the dimensionality. These methods are mostly used in robotics and they are the basis in generating robot motion for a large panel of applications and tasks.

Yet, there are many tasks where the robot is exploited to its extreme dynamics, capabilities and performances. Examples of which are ultra-precise operating motion in industries, or kicking motions [1], [2] for humanoid robots or fast motion with collision avoidance for manipulator robot [3] and lifting heavy objects [4]. These extreme tasks are impossible to achieve by state-of-the art planning and task-based closed-loop control. Generating off-line explicit and optimal trajectories is most often the only and the safest option.

In these cases, motion planning is an optimal control problem since it consists in finding the best continuous joint trajectories that minimize a cost function and satisfy a set of constraints all along the space and time domains. Using a joint trajectory parameterization reduces this Infinite Programming problem into a Semi-Infinite Programming one (SIP). SIP aims at finding the best finite set of parameters, describing the joint trajectories, which satisfy the continuous constraints while minimizing the cost function. To solve this SIP problem using state-of-the-art optimization methods, we still need to express the continuous constraints into a finite set of discrete constraints. We can either use a time-grid [5] or time-interval [6] discretization of the continuous constraints. The first one quickly produces a solution to the problem but does not guarantee any constraint satisfaction in-between time samples; the second one does but at the cost of non-negligible computation time. Besides, neither the time-interval nor the time-grid discretization allows taking into account equality constraints all along the motion duration.

First, the bounds on the joint values and velocity (even acceleration) are checked without time discretization; this is made by taking advantage of some B-Splines properties.

But our contribution is about decreasing the computation time of functions over time-intervals using Taylor polynomials with interval remainder bounds [7]. This technique has been used in the context of continuous collision detection process in computer graphics [8]. Our extension to robotics allows dealing with sets of entire motion inequality and equality constraints, maximal derivative constraints, and gives an easy way to compute the objective function.

We assess our method on the HRP-2 robot by computing a stable dynamic kicking motion while keeping the right hand at a constant position all along the motion. The remaining of the paper describes what the method is about, its implementation, performances and obtained experimentation results.

I. PROBLEM STATEMENT

A. Semi-Infinite Programming

We aim at obtaining whole-body joint trajectories that achieve task objectives that can be expressed as a set of constraints [9] while minimizing a cost function for an anthropomorphic system such as the Humanoid Robot HRP-2 [10]. Thus, we have to find the best joint trajectories $q(t)$ that solve the following infinite problem:

$$\begin{array}{ll} \underset{q(t)}{\operatorname{argmin}} & C(q(t)) \\ \forall i, \forall t \in [0, T_f] & g_i(q(t)) \leq 0 \\ \forall j, \forall t \in [0, T_f] & h_j(q(t)) = 0 \\ \forall t_k \in \{t_1, t_2, ..., t_n\} & z_k(q(t_k)) \leq 0 \end{array} \quad (1)$$

where g, h are the sets of continuous inequality and equality constraints which must hold throughout the motion duration, z is a set of the discrete inequality constraints which must hold only for a discrete set of instants, C is the cost function to minimize. In this paper, we emphasize on the constraints computation. Yet, our method can also deal with the computation of the cost function (Section IV-C).

To reduce the complexity of the problem so that it is computationally solvable we define a parameter set $\mathbf{X} \in \mathbb{R}^N$ that defines the joint coordinates $q(t)$. We choose B-Splines curves parameterization for the joint coordinates which results in a finite set of control points. These control points constitute our new set of optimization parameters, as previously done in e.g. [11], [2]. The B-Splines properties will be discussed in Section II, and the motion planning problem reduces into finding the best parameter set $\mathbf{X} \in \mathbb{R}^N$ that:

$$\begin{array}{cc} \underset{\mathbf{X}}{\text{argmin}} & C(\mathbf{X}) \\ \forall i, \forall t \in [0, T_f] & g_i(\mathbf{X}, t) \leq 0 \\ \forall j, \forall t \in [0, T_f] & h_j(\mathbf{X}, t) = 0 \\ \forall k & z_k(\mathbf{X}, t_k) \leq 0 \end{array} \tag{2}$$

This problem turns to a Semi-Infinite Programming one (SIP) [12] since it deals with a finite set of parameter $\mathbf{X} \in \mathbb{R}^N$ yet with continuous constraints that can be decomposed into an infinite set of discrete constraints. Let us briefly recall some different options to solve SIP.

B. How to solve SIP

In most cases, SIP is solved using a time-grid discretization of the constraints [12], [13], [5]. Thus, the inequality constraints in the problem (2) are replaced by:

$$\forall i, \forall t_d \in \mathbb{T} \quad g_i(\mathbf{X}, t_d) \leq 0 \tag{3}$$

where $\mathbb{T} = \{t_s, t_1, ..., t_e\}$ is the time-grid used for the discretization process; the constraints will be checked only for the discrete instants in $\mathbb{T}$. As discussed in [1], [6], it is not easy to guess a time-grid discretization which guarantees that the constraint also holds in-between a pair of time-point of the grid. It is also difficult to compromise performance between having a reasonable sampling and constraint satisfaction of predefined single time constraints of type z. Those are the reasons why we use interval discretization which replaces the inequality constraint in (2) by:

$$\forall i, \forall [t] \in \mathbb{IT} \quad \sup [g]_i(\mathbf{X}, [t]) \leq 0 \tag{4}$$

where $\mathbb{IT} = \{[t_0], [t_1], ...\}$ is a set of time-interval that covers the entire motion duration $\forall t \in [0, T_f], t \in \mathbb{IT}$. Unfortunately, this method, which guarantees constraints satisfaction all along the motion, is computationally very expensive because it uses a bisection process to get tight enclosures of extrema. To have an idea about the order of complexity, it took more than twenty hours to compute the motion planning of a humanoid robot of twelve degrees of freedom for a free kicking task [1]. Besides, neither the time-interval nor the time-grid discretization methods can deal efficiently with equality constraints $\forall j, \forall t \in [0, T_f]\ h_j(X, t) = 0$.

We propose a new method which avoids joint position and velocity discretization using a particular B-Splines property and uses Taylor polynomial expression to get tight enclosures in small CPU-time for the other constraints.

II. Joint limits constraints

We present a method to enforce joint position, speed, and acceleration bounds (i.e. limits) without any discretization. To do so, we make use of some niceties from B-Splines' properties [14].

A. Definition and convex hull property

A B-Splines function is the weighted sum of basis function defined by m control points and K is the order of the basis functions.

$$S(t) = \sum_{i=1}^{m} b_i^K(t) p_i \tag{5}$$

A B-Splines curve is entirely contained in the convex hull of its control polyline. This property is obtained quite easily from the definition of the basis functions as follows:

$$\forall t \in [0, T_f] \quad \textstyle\sum_{i=1}^{m} b_i^K(t) = 1 \tag{6}$$

This immediately yields:

$$\forall i \in [1, m] \quad \underline{S} \leq p_i \leq \overline{S} \Rightarrow \forall t \in [0, T_f] \quad \underline{S} \leq S(t) \leq \overline{S} \tag{7}$$

We can simply use this property to map joint limits to optimization parameters bounds, thus avoiding the need to implement these limits in terms of time-discrete set of inequality constraints.

B. Derivative of a B-Spline

The time-derivate of our B-Spline defined with m control points is another B-Spline parameterized with $m-1$ control points and of $K-1$ degree. This is obtained by derivation of the Cox-de-Boor recursion [14] with respect to time t, that is:

$$\dot{S}(t) = \sum_{i=1}^{m} \dot{b}_i^K(t) p_i = \sum_{i=1}^{m-1} b_i^{K-1}(t) r_i \tag{8}$$

with:

$$r_i = \frac{K}{u_{i+K+1} - u_{i+1}} (p_{i+1} - p_i) \tag{9}$$

here u_i is the i^{th} component of the nodal vector as defined in [14]. It is then possible to obtain a system of $(m-1)$ linear inequalities to impose joint speed limits, in the same way as we can enforce joint position limits inequality constraints as bounds on the B-Splines parameters.

Thus we add the following constraints:

$$\underline{\dot{q}} \leq r_i \leq \overline{\dot{q}} \tag{10}$$

Extensions to upper derivative follow the same principle.

C. Optimality

Note that by doing so, the obtained solution might be sub-optimal as there is no reciprocal to Equation (7), thereby reducing the size of the feasible set of parameters. This can be mostly resolved by increasing the number of control points, and will be good enough for fairly smooth motions.

III. Interval Analysis and Taylor polynomials

Using the properties of the B-Splines functions, we are now able to take into account the constraints on the joint position and derivatives. Unfortunately, the continuous constraints on the balance and the joint torques cannot be simplified the same way. Thus, we propose to compute the maximum of this continuous inequality constraints over time-interval as we did in a previous paper [1], but in this case to reduce the overestimation of the function we use a Taylor polynomial.

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 was extended to validated numerics [15], [16], [17].

A real interval $[a] = [\underline{a}; \overline{a}]$ is a connected and closed subset of $\mathbb{R}$. With $\underline{a} = \inf([a])$, $\overline{a} = \sup([a])$ and $\text{mid}([a]) = \frac{\underline{a}+\overline{a}}{2}$. The set of all real intervals of $\mathbb{R}$ is denoted by $\mathbb{IR}$. Real arithmetic operations are extended to intervals. Consider an operator $\circ \in \{+, -, \times, \div\}$ and $[a]$ and $[b]$ two intervals. Then:

$$[a] \circ [b] = [\inf_{u\in[a],v\in[b]} u \circ v, \sup_{u\in[a],v\in[b]} u \circ v] \tag{11}$$

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

B. Taylor Polynomials

One major drawback of Interval Analysis is the computation time which is due to the overestimation of the function. This overestimation produces a conservative interval that contains the actual solution interval but is too large to be used. In [1], we use a bisection process that cuts the interval into several sub-intervals and computes the interval result as the union of all the sub-interval results. We applied this method to motion planning of a 12-dof humanoid robot; such a process resulted in cutting into 2^{10} subintervals which produced a computation time superior to 20 hours [1].

In this paper, we reduce the computation time by using Taylor polynomials. A major reason for overestimation during an interval computation is that Interval Analysis does not keep track of the correlation between the different sub-functions [8]. That is why we propose to define each function over a time interval $[t_s, t_e]$ as the sum of a $n-$order polynomial function and an error interval:

$$\forall t \in [t_s, t_e] \;\; f(t) \in \sum_{i=0}^{n} a_i \times t^i + [\varepsilon] \tag{12}$$

where $\{a_0, a_1, \ldots, a_n\} \in \mathbb{R}^{n+1}$ are the coefficient of the polynomial and $[\varepsilon] \in \mathbb{IR}$ is the error interval remainders bounds. To compute the inverse dynamic model of the robot, we need to implement the result of a few operations (sum, subtraction, multiplication, sine and cosine trigonometric functions) as the sum of a polynomial and an error interval. To do so, we implemented the computation presented in [7].

C. Computing extrema

Now that we are able to compute all the functions over a time interval $[t_s, t_e]$ as the sum of a $n-$order polynomial function and an error interval, we want to evaluate, in a small CPU-time, the extrema of this function. To do so, we propose to use the property (7) of the B-Splines explained in section II. Given a polynomial:

$$P(t) = [a_0, a_1, \ldots, a_N] \times \left[1, t, \ldots, t^N\right]^T \tag{13}$$

and knowing the coefficients a_i, we want to compute the coefficients p_i of the equivalent B-Splines function:

$$P(t) = [p_0, p_1, \ldots, p_N] \times \mathbf{B} \times \left[1, t, \ldots, t^N\right]^T \tag{14}$$

where $\mathbf{B}$ is a matrix that contains the polynomial parameters of the B-Splines basis functions. Note that the basis functions used to evaluate extrema are different from the basis function used to define the motion. Therefore, we can compute the corresponding B-Splines parameters such that:

$$[p_0, p_1, \ldots, p_N] = [a_0, a_1, \ldots, a_N] \times \mathbf{B}^{-1} \tag{15}$$

The Matrix $\mathbf{B}$ relies on the order of the Taylor approximation and on the time interval $[t_s, t_e]$, thus we need to compute it only once at the beginning of the optimization process. In addition, we are able to compute inferior and superior bounds for any function (torque, ZMP, etc.) thanks to:

$$\begin{array}{c} \min_{t\in[t_s,t_e]}(f(t)) \geq \min_i(p_i) + \inf([\varepsilon]) \\ \max_{t\in[t_s,t_e]}(f(t)) \leq \max_i(p_i) + \sup([\varepsilon]) \end{array} \tag{16}$$

IV. Additional Applications

Initially, we implemented the Taylor polynomials to deal with the computation of the maximum of inequality constraints over the time-interval. But, this tool can also be very useful to evaluate continuous equality and maximal variation constraints and to compute the cost function as well.

A. Equality constraints

Since we have a polynomial expression of each value of the robot, we can add some constraints on the parameters of the polynomial to define an equality constraint:

$$\forall i \in \{1, \cdots, N\} \quad a_i = 0 \tag{17}$$

$$a_0 = \text{const.} \tag{18}$$

This set of constraints gives the opportunity to get a constant function (equal to a defined or not value) all along the motion. With classical methods, the set of equality constraints $\forall j, \forall t \quad h_j(X,t) = 0$ is discretized using a time-grid, and the equalities are verified only for the time-grid values. With our method we guarantee that the equality holds for the entire time-space motion. Subsequently, we can now set, as a constraint, a constant position of any robot body, which is for example useful for multi-contact motions.

B. Maximum of variations

It can be of some use to add constraints on the derivative of any functions $\mathbf{f}$, that is:

$$\underline{\Delta} \leq \frac{d\mathbf{f}(t)}{dt} \leq \overline{\Delta} \tag{19}$$

Since we have a polynomial expression of the functions $\mathbf{f}$, we can easily differentiate it with respect to time to get another polynomial expression of the derivative, that is:

$$\frac{d\mathbf{f}(t)}{dt} = \sum_{i=1}^{N} i \times a_i \times t^{i-1} \tag{20}$$

Then it is easy using Eq. (20) to compute the extrema of the variations over time interval.

C. Objective Function

The polynomial computation of the functions can also be used for computing cost functions. In most cases, the objective function is the result of the integration of a function $\mathbf{f}(t)$ (which can be the sum of the square torques, jerks...) all along the motion duration.

$$C(\mathbf{X}) = \int_{t_s}^{t_e} \mathbf{f}(t)dt \tag{21}$$

The objective value is usually obtained by a discrete integration, but using the polynomial expression we can compute the objective function as:

$$C(\mathbf{X}) = \sum_{i=0}^{N} \frac{a_i}{i+1}(t_e^{i+1} - t_s^{i+1}) \tag{22}$$

Now that we have settled all the ingredients, we highlight the performance of our approach in a robotic experimental example.

V. Experimental Validation

We applied the method presented in this paper to optimize a kicking motion with a humanoid robot HRP-2 [10]. This task was chosen because it has been performed in [2] using a time-grid discretization; so we have a good mastering of the robot behavior for this task. Moreover it is a full body motion; that involves all the joints of the robot. It also uses one contact support which allows not taking in account internal forces due to multi-contact supports (our on-going work).

A. Modeling

To model the motion of the robot we make two assumptions that we make plausible by enforcing the optimization process with appropriate additional constraints. First, we assume that the contact between the HRP-2 supporting foot (the right one) and the ground always holds. Subsequently it can be considered as a bilateral contact, so the right foot does not change position and orientation during the motion. Both of those assumptions are enforced by taking into account some constraints during the optimization process.

The purpose of this assumption is to define the humanoid robot HRP-2 as tree chain with the right foot as the reference body. Hence, knowing the joint value velocity and acceleration, we can compute the joint torques (Γ) and the contact/interaction force between the reference body and the ground (F_{ref}) thanks to the following equation:

$$[\Gamma(t), \mathbf{F}(t)]^T = \mathbf{M}(q(t))\ddot{q}(t) + \mathbf{H}(q(t), \dot{q}(t)) \tag{23}$$

where $\Gamma(t)$ are the joint torques, $\mathbf{F}(t)$ the six-component of the foot/ground contact force, $\mathbf{M}(q)$ the inertial matrix and $\mathbf{H}(q, \dot{q})$ the vector of the Coriolis and gravity forces. No other external force is considered for this experiment.

Assuming a bilateral contact implies that there is no sliding, no take-off, and no turn over the edges of the foot in contact. This assumption is enforced through explicit constraints expressing such conditions, namely:

$$\forall t \in [0,T] \begin{cases} F_x(t)^2 + F_y(t)^2 \leq \mu F_z(t)^2 \\ F_z(t) \geq 0 \\ \underline{ZMP_s} \leq ZMP_s(t) \leq \overline{ZMP_s} \\ \underline{ZMP_f} \leq ZMP_f(t) \leq \overline{ZMP_f} \end{cases} \tag{24}$$

The first equation translates the constraint of no sliding through the Coulomb friction law and expresses that the contact force remains within the friction cone (define by μ), the second one expresses no taking off, and the two last are about the balance to avoid turning over the edges of contact: the ZMP is the Zero Moment Point as recently revisited in [18] and is computed using the contact forces.

Our second assumption is to consider the robot as poly-articulated rigid bodies, though in reality there are some flexibility effects due mainly to the compliance of the absorbing choc mechanism embedded in the ankle. S. Kajita and his colleagues designed a specific controller to handle humanoid's flexibilities of the ankle and hence reduce the compliance effects. Nevertheless, these controllers are efficient only for reasonably small variations of the ZMP [19], [20].

Therefore, we added constraints on the ankle torque variation to minimize the effect of the compliance during the motion. To do so, we used the result of section IV-B: we have a polynomial expression of the functions that we derivate with respect to time and compute the extrema variation over time interval, see Eq. (20).

We also take into account constraints on the joint position, velocity and torques:

$$\forall t \in [0,T] \begin{cases} \underline{\Gamma} \leq \Gamma(t) \leq \overline{\Gamma} \\ \underline{q} \leq q(t) \leq \overline{q} \\ \underline{\dot{q}} \leq \dot{q}(t) \leq \overline{\dot{q}} \end{cases} \tag{25}$$

B. Optimization problem

We choose to compute the robot's dynamic motion trajectories by setting 14 control points per motorized joint. We also set initial and final joint's velocities and accelerations to zero; this means that among the 14 control points, 4 are already set by the initial conditions and 10 remain variable. Subsequently, we create an optimization with $(30 \times 10 + 1) = 301$ parameters (we add the motion duration) that has to minimize the sum of the square torque change (computed with the equations (20) and (22)). Constraints on the joint

values and velocities are expressed with the method presented in section II. We use a polynomial computation to ensure maximal torque values, maximal torque change and the constraint presented in Eq. (24) due to the assumption of bilateral contact. In addition, we define a set of task equality constraints h_j to ensure that the position and the orientation of the HRP-2 right gripper remains in a set constant position and orientation all along the kicking motion. We also add discrete constraints z_k to specify the behavior of the robot (position and velocity of the foot at the mid-duration...). We set the velocity of the flying foot (left) to 3.0m/s at the mid-duration of the motion (when impact occurs). All these supplementary constraints aim to cover all types of constraints and to add complexity to the motion generation relatively to [2].

C. Constraint computation

The number of control points suggests splitting the motion duration into ten intervals. Table I shows the computation time of all the constraints for different order of the polynomials, with and without the computation of the remainder error interval $[\varepsilon]$. This computation time is for one step of the optimization process (it does not mean the convergence time of the optimization process which is around one hour of computation time).

TABLE I

COMPUTATION TIMES OF ALL THE CONSTRAINTS.

polynomial order	computation of $[\varepsilon]$	CPU time (ms)
5	no	310
10	no	560
5	yes	530
10	yes	980

Figure 1, shows the torque of the hip and its computation using the sum of a 5^{th}-order Taylor polynomial and the interval error. We can see that the error is not negligible. Nevertheless, it appears that the Taylor polynomial is very close to the actual value of the torques. Using a 10^{th}-order Taylor polynomial we guess that the error is very small and that the Taylor polynomial is very closed to the actual function.

Moreover, Fig. 2 shows the computation of the extrema over the ten time-intervals using a 5^{th}-order Taylor polynomial and a 10^{th}-order Taylor polynomial with and without computing the error. We can see that the 5^{th}-order computation produces nearly the same results as the 10^{th}-order computation. Thus to have fast computation, it is possible to use a 5^{th}-order Taylor polynomial without computing the error, but to have safe motion we use a 10^{th}-order Taylor polynomial with error computation.

D. Optimal motion

Table II contains the computation time for motion planning using both configurations. The optimization process gives nearly the same results, since the cost functions are very close to each other. As expected, the method which computes

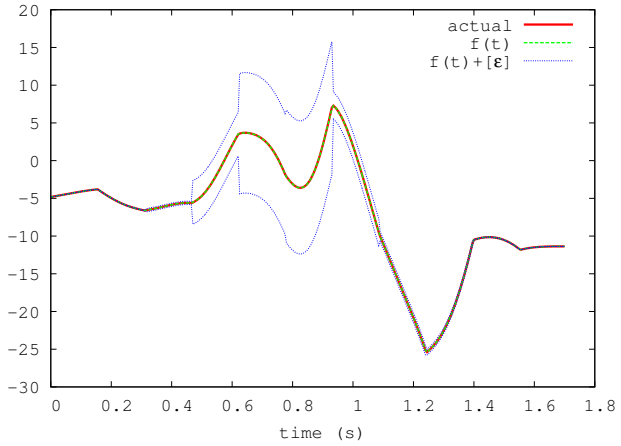

Fig. 1. Torque of the hip (Nm). Computation with a 5^{th} order polynomial.

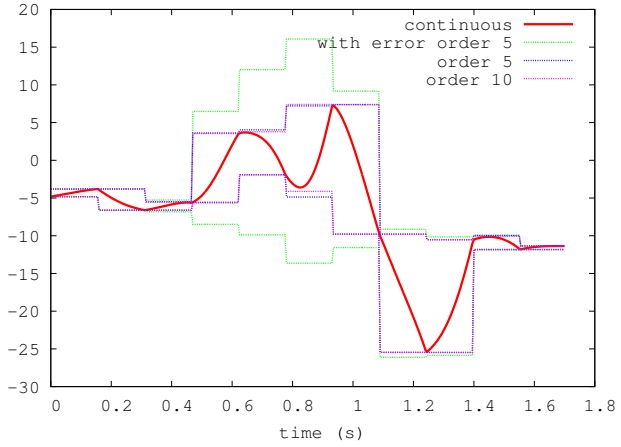

Fig. 2. Computation of the extrema for the torque of the hip (Nm).

the remainder error interval is slower, but it better enforces the validity of the constraints along the motion.

TABLE II

MOTION PLANNING CPU-TIME.

type	5^{th}-order without $[\varepsilon]$	10^{th}-order with $[\varepsilon]$
iterations	256	273
cost function	$1.29e^5$	$1.19e^5$
CPU time (mn)	59	234

Even if there is a considerable gain on the variable size, we have a longer computation time relatively to methods using classical time-grid discretization that do not produce safe motions. But, comparing to previous safe motion planning method presented in [1] that produces a 12-dof motion in more than twenty hours, this new method is considerably faster since it produces a motion for the 30-dof robot within one hour.

E. Experiments

Figure 3 shows the experiment where the HRP-2 robot is kicking a ball, while he has a ball on the right hand to

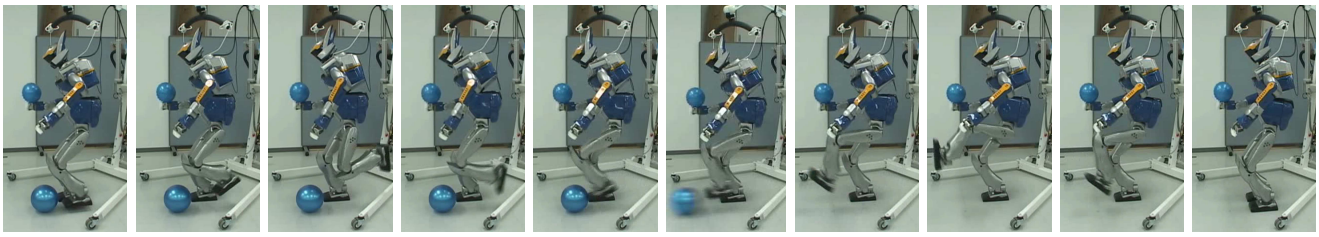

Fig. 3. Kicking motion: we put a light ball on the right gripper to highlight the constant position and orientation of the right hand. The ball does not fall during the kicking motion.

highlight the constant position and orientation of the right hand. During the optimization process we did not take into account the balls' weights. Indeed, the ball put on the gripper is light enough (93 grams), to not make the robot fall, and the second one (to be kicked) is soft and light enough (105 grams) to not consider switching to impact models. We ran three attempts with this configuration, two of them make the ball of the hand fall, one, presented on Fig. 3 makes the ball remain on the hand. During modeling, we consider rigid body without any flexibility. The stabilizer changes joint values independently to counterpart the effect of the flexibility to ensure the balance of the robot despite keeping the constant position of the hand. To avoid this phenomenon, we plan to add a task to avoid the motion of the hand in the controller of the robot.

This experiment validates the motion computed off-line, since the robot does not fall, even if additional technical works is needed to improve the reproducibility.

Conclusion

We addressed motion generation using optimization techniques and focused on the computation of the constraint function for SIP. Most of methods compute the constraint functions over a time-grid. We explained how constraints on the joint position and velocity can be implemented without any discretization by taking into account constraints on the B-Splines control points. For other constraints (joint torques, balance...) we propose a method to compute the constraint over time-intervals as the sum of a Taylor polynomial approximation and an error interval, which makes possible to evaluate easily the extrema of any constraints and their derivatives. Moreover, we used the Taylor polynomial to take into account continuous equality constraint, continuous change inequality constraint, and to compute the cost function. We validate this method by computing a kicking motion for the humanoid robot HRP-2. The computation time we have for this example are better than previous safe motion planning methods, but still to be improved relatively to classical time-grid discretization.

As future works, we will investigate on decreasing the computation time and we will also extend the example to multi-contact motion [9] and collision avoidance [8].

References

[1] S. Lengagne, N. Ramdani, and P. Fraisse, "Planning and fast replanning of safe motions for humanoid robots : Application to a kicking motion," in *IROS 2009*, p. 441446.

[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 *ROBIO 2006*, pp. 299–304.

[3] A. I. F. Vaz, E. M. Fernandes, and M. P. S. Gomes, "Robot trajectory planning with semi-infinite programming," *European Journal of Operational Research.*, vol. 153, no. 3, pp. 607–617, 2004.

[4] H. Arisumi, S. Miossec, J.-R. Chardonnet, and K. Yokoi, "Dynamic lifting by whole body motion of human robots," in *IROS*, september 2008, pp. 668–675.

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

[6] S. Lengagne, N. Ramdani, and P. Fraisse, "Safe motion planning computation for databasing balanced movement of humanoid robots," in *ICRA 2009*, pp. 1669–1674.

[7] M. Berz, G. Hoffsttter, and G. H. Atter, "Computation and application of taylor polynomials with interval remainder bounds," *Reliable Computing*, vol. 4, pp. 83–97, 1998.

[8] X. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using taylor models and temporal culling," *ACM Trans. Graph.*, vol. 26, no. 3, p. 15, 2007.

[9] A. Escande and A. Kheddar, "Contact planning for acyclic motion with tasks constraints," in *IROS 2009*, Oct. 11-15.

[10] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *ICRA 2004*, vol. 2, Apr./May, pp. 1083–1090.

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

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

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

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

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

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

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

[18] 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.

[19] S. Kajita, K. Yokoi, M. Saigo, and K. Tanie, "Balancing a humanoid robot using backdrive concerned torque control and direct angular momentum feedback," in *ICRA 2001*, vol. 4, pp. 3376–3382.

[20] S. Kajita, T. Nagasaki, K. Kaneko, K. Yokoi, and K. Tanie, "A running controller of humanoid biped HRP-2LR," in *ICRA 2005*, Apr., pp. 616–622.