Using a Multi-Objective Controller to Synthesize Simulated Humanoid Robot Motion with Changing Contact Configurations

Karim Bouyarmane and Abderrahmane Kheddar

***Abstract*— Our objective in this work is to synthesize dynamically consistent motion for a simulated humanoid robot in acyclic multi-contact locomotion using multi-objective control. We take as an input a planned sequence of static postures that represent the contact configuration transitions; a multi-objective controller then synthesizes the motion between these postures, the objectives of the controller being decided by a finite-state machine. Results of this approach are presented in the attached video in the form of playback motions generated through non-real-time constraint-based dynamic simulations.**

I. Introduction

In our previous work [1], [2] we presented an algorithm that plans a sequence of multi-contact stances with corresponding static postures that brings a humanoid robot from an initial configuration to a desired stance/configuration. As opposed to the walking pattern generation problem, this approach is aimed at generating non-gaited acyclic motion with arbitrary contact configurations (using hands, forearms, knees, etc.). The presented algorithm was the first of a two-stage contact-before-motion planning framework. The second stage, which is the main concern of the present paper, is to synthesize a continuous motion that goes through the planned sequence of static postures. Previous approaches of the problem [3], [4] used randomized motion planning techniques (RRTs, PRMs) to plan the continuous motion. However, due to the geometric nature of such techniques, which is not suited for the integration of dynamics motion constraints in the planning, their motion was restricted to be quasi-static, meaning that static equilibrium is respected at every time of the motion. Adapting kinodynamic planning or dynamic filtering techniques [5], [6] to the output motion of these works could have been one way to overcome this limitation.

In this paper we investigate a different approach, that has both its advantages and drawbacks over the previous one. On the plus side it directly synthesizes dynamically consistent motion, for which the dynamics equation of motion is satisfied throughout the motion. The main drawback of this approach is that it does not allow for explicit formulation of collision-freeness constraint, which induces us to resort to hand-designed heuristics in order to avoid collisions. However though, the closed-loop nature of our approach makes it robust to unavoided collision (contact) events that may occur during the generation of the motion, as these collisions are seen as perturbations "absorbed" by the feedback motion generation law.

The organization of the paper is as follows. After discussing related work (Section II) and an overview of the approach (Section III), we get to the detailed technical developments by first presenting the multi-objective controller used for a single step motion (Section IV) followed by the finite-state machine used for multiple steps motion (Section V). Finally we describe the results that appear in the attached video (Section VI).

II. Related Work and Contribution

The method we choose is inspired by recent trend in computer graphics community, synthesizing physics-based motion of simulated characters [7], [8]. They formulate the motion generation problem as the control problem of the human character within simulation. Humanoid robotics [9], [10] as well as virtual reality communities [11] recently started using same/similar formulations in their applications.

[7], [9] use motion-capture data to generate the motion, their objective being precisely to track these data with the simulated human character or humanoid robot. Our method here does not need such data as it relies only on the information carried by the first stage of the contact-before-motion planning framework that produces the sequence of static postures. This sequence of static postures plays the same role the motion-capture data does in the other works, i.e. solving for redundancies and producing natural looking motion. Therefore one of the main original features in our method is increased autonomy of the robot. [8] also does not rely on motion-capture data, but their applications are restricted to human cyclic walking and a single upstanding posture is sufficient to solve for the said redundancies, while we are targeting more general acyclic motions.

Moreover, most of these works [7], [9], [11], [10] produce motion for the robot in a single stance, standing either on one foot or two-feet stances, without changing their contact configuration. As in [8] our method adds a finite-state machine to perform motions that go through changing contact-configuration stances, but, once again as opposed to [8], in an acylic way.

Note, however, that although [7], [8] report computation times that reach near real-time objectives using an LCP-based simulator on the animated characters, we do not focus in this work on optimizing the computation time to be real-time since in our current implementation some real robot constraints (self-collisions and joint limits) need to be checked a posteriori, i.e. once the full motion has

The authors are with CNRS-AIST JRL (Joint Robotics Laboratory) UMI3218/CRT, AIST, Tsukuba, Japan; and with CNRS-University of Montpellier 2 LIRMM, Montpellier, France. {karim.bouyarmane,abderrahmane.kheddar}@aist.go.jp

been generated, before safely executing it on the robot (cf. Section VI). This is why our method here is presented as an off-line motion generation tool in simulation rather than an on-line control one directly embedded in the real robot.

Other approaches use different formulations to control/generate motion of humanoid robot in multi-contact stances [12], [13]. The former uses a prioritized tasks hierarchy formulation but does not explicitly take into account contact constraints, the latter approach is conceptually different as it generates optimal motion through a semi-infinite optimization formulation of B-spline parametrized motion. While no time complexity analysis has been reported for [12], the approach in [13] applied to humanoid robot requires for now computation times as high as a few hours to generate a one-minute motion.

III. Overview of the Method

Fig. 1 shows an overview of the proposed motion generation method. In this figure the multi-contact stances planner block [2] and the simulator block [14] are considered as black-box modules. Their implementation is not developped in this paper, which focuses on the finite-state machine and the multi-objective controller designs. t denotes the simulation time; q, $\dot{q}$, and $\ddot{q}$ denote respectively the configuration, configuration velocities, and configuration accelerations of the humanoid robot, which include both the actuated joints and the root $SE(3)$ component of the robot; f denotes the aggregated vector of contact forces applied at finite contact points between the robot and the environment; and u denotes the actuator torques that control the simulated robot. q_s and q_g are the start and goal configuration input by the user. The motion parameters, also input by the user, include the step time, step height, etc. and are further detailed in the finite-state machine description section (Section V).

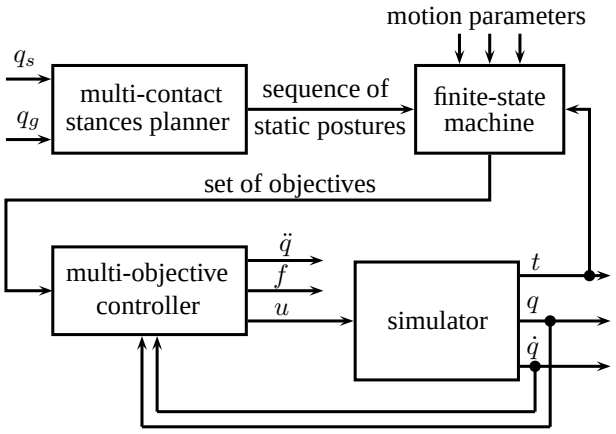


Fig. 1. Overview of the motion generator

At every time step of the simulation, the finite-state machine decides on the objectives to feed to the controller, which then uses a quadratic formulation that solves for the configuration accelerations, the contact forces, and the control torques. Note that the produced configuration accelerations $\ddot{q}$ could be directly integrated to update q and $\dot{q}$. We choose however to discard the produced $\ddot{q}$, along with the computed contact forces f, and to keep only the control u that we feed to the simulator which will in turn output a more accurate f and $\ddot{q}$ to be integrated. This approach, the same as the one chosen in [7], [8], allows for a complete decoupling of the controller and the simulator blocks, the latter can later be replaced by any other one, more accurate or faster, depending on the targeted application. In particular, replacing the simulator by the real robot can be seen as particular case, provided that adequate sensors/estimators feed us back with q and $\dot{q}$ (especially the $SE(3)$ components of these vectors).

IV. Multi-Objective Controller

The multi-objective controller minimizes a weighted sum of objectives subject to the following constraints:

- satisfy the dynamics equation of motion,
- non-sliding of the contact points,
- contact forces inside the (linearised) friction cone,
- actuation torques within their limits.

The objectives are specified in terms of *tasks* (the term *features* is alternatively used in the literature). A task is for example driving the position of an end-effector, the center of mass of the robot, the whole configuration of the robot, etc. Conflicts between different tasks are solved through a weighted formulation rather than strict prioritization. More technical details of the optimization problem formulation is found in the following subsections:

A. The Linear Constraints

Let us suppose we have a humanoid robot made of r revolute joints and $r+1$ links indexed by $k \in \{0, \dots, r\}$. On each link k a set of contact forces $f_{k,1}, \dots, f_{k,m_k}$ are applied at the respective local-frame-expressed points $a_{k,1}, \dots, a_{k,m_k}$. Let $q = (x_0, \theta_0, \hat{q}) \in \mathbb{R}^{3+4+r}$ denote the configuration vector of the humanoid robot, where x_0 is the global-frame-expressed position of the root, θ_0 a parametrization of its orientation (a unit quaternion for instance), and $\hat{q}$ the internal (actuated) joint angles vector. For each k, $J_{tk}(p)$ denotes the $3 \times (3+4+j)$ translational Jacobian of the link k relative to the global frame with respect to q expressed at a local-frame-expressed point p.

The motion of the humanoid robot is governed by the following equation (see [15] from which we borrow the notations for details on how we derive this equation, especially for the expressions of M and N, g is the gravity vector):

$$M(q)\ddot{q} + N(q,\dot{q})\dot{q} = M(q) \begin{pmatrix} g \\ 0_4 \\ 0_r \end{pmatrix} + \begin{pmatrix} 0_3 \\ 0_4 \\ u \end{pmatrix} + \sum_{k,i} J_{tk}(a_{k,i})^T f_{k,i}\,, \tag{1}$$

The motion is additionally subject to the following constraints, denoting $K_{k,i}$ the Coulomb friction cone at $a_{k,i}$,

$$\forall\, k, i \quad J_{tk}(a_{k,i})\,\dot{q} = 0\,, \tag{2}$$

$$\forall\, k, i \quad f_{k,i} \in K_{k,i}\,, \tag{3}$$

$$\forall\, j \quad u_{j,\min} \leq u_j \leq u_{j,\max}\,. \tag{4}$$

We linearise the friction cone $K_{k,i}$ by specifying a finite set of global-frame-expressed generators $\{v_{k,i,1}, \ldots, v_{k,i,\nu_{k,i}}\}$ so that each contact force $f_{k,i}$ is a non-negative linear combination of the vectors v:

$$f_{k,i} = \sum_{\mu=1}^{\nu_{k,i}} \lambda_{k,i,\mu}\, v_{k,i,\mu}\,, \tag{5}$$

$$\forall\, k, i, \mu \quad \lambda_{k,i,\mu} \geq 0\,. \tag{6}$$

We denote $\lambda = (\lambda_{k,i,\mu})_{k,i,\mu}$.

By time-differentiating the constraint (2) we get

$$J_{tk}(a_{k,i})\,\ddot{q} + \dot{J}_{tk}(a_{k,i})\,\dot{q} = 0\,. \tag{7}$$

Let us define the parameter vector $X = (\ddot{q}, \lambda, u)$. For clarity we denote

$$\alpha = \dim(\ddot{q}) = 3 + 4 + r\,, \qquad \gamma = \dim(u) = r\,, \tag{8}$$

$$\beta = \dim(\lambda) = \sum_{k=0}^{m}\sum_{i=1}^{m_k} \nu_{k,i}\,, \qquad \zeta = \sum_{k=0}^{m} m_k\,. \tag{9}$$

Furthermore, for a family of same-size matrices $(Y_\iota)_{\iota\in\{1,\ldots,I\}}$, we denote the block aggregation operators

$$\lfloor Y_\iota \rfloor_{\iota\in\{1,\ldots,I\}} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_I \end{pmatrix}\,, \quad \lceil Y_\iota \rceil_{\iota\in\{1,\ldots,I\}} = \begin{pmatrix} Y_1 & \ldots & Y_I \end{pmatrix}\,. \tag{10}$$

Finally, the equation of motion (1) and constraints (4), (6), (7) take the following linear form

$$A_1\, X = B_1\,, \quad A_2\, X \leq B_2\,, \tag{11}$$

where the matrices A_1,A_2 and the vectors B_1,B_2 are defined

$$A_1 = \begin{pmatrix} M(q) & -\lceil J_{tk}(a_{k,i})^T v_{k,i,\mu} \rceil_{k,i,\mu} & -\begin{pmatrix} 0_{3\times\gamma} \\ 0_{4\times\gamma} \\ 1_{\gamma\times\gamma} \end{pmatrix} \\ \lfloor J_{tk}(a_{k,i}) \rfloor_{k,i} & 0_{3\zeta\times\beta} & 0_{3\zeta\times\gamma} \end{pmatrix}\,,$$

$$B_1 = \begin{pmatrix} -N(q,\dot{q})\dot{q} + M(q)\begin{pmatrix} g \\ 0_4 \\ 0_r \end{pmatrix} \\ \lfloor -\dot{J}_{tk}(a_{k,i})\dot{q} \rfloor_{k,i} \end{pmatrix}\,,$$

$$A_2 = \begin{pmatrix} 0_{\beta\times\alpha} & -1_{\beta\times\beta} & 0_{\beta\times\gamma} \\ 0_{\gamma\times\alpha} & 0_{\gamma\times\beta} & -1_{\gamma\times\gamma} \\ 0_{\gamma\times\alpha} & 0_{\gamma\times\beta} & 1_{\gamma\times\gamma} \end{pmatrix}\,, \quad B_2 = \begin{pmatrix} 0_\beta \\ -u_{\min} \\ u_{\max} \end{pmatrix}\,. \tag{12}$$

Let us now write the target function to optimize.

B. The Quadratic Objectives

We define a *task* (or *feature*) as a scalar or vector function g of the configuration of the robot $g : \mathbb{R}^{3+4+r} \to \mathbb{R}^d$, where d is the dimensionality of the task. Example of such tasks include the global-frame expression of a particular point attached to one of the robot's links ($d = 3$), the CoM of the entire robot ($d = 3$), the configuration itself of the robot ($d = 3 + 4 + r$), etc. Let J_g denote the Jacobian of the task, i.e. the $(3 + 4 + r) \times d$ matrix $J_g(q) = \partial g/\partial q$.

As proposed in [8] we will use two kinds of objectives for the task g:

- a *set-point objective*, denoted $E_{\mathrm{spt},g}$, used if we wish to servo the task g around an given reference value g_{ref},
- a *target objective*, denoted $E_{\mathrm{tgt},g}$, used if we wish to steer the task g from a given initial value $(g_0, \dot{g}_0)$ to a given target final value $(g_f, \dot{g}_f)$ in given time t_f.

The Set-point Objective: The corresponding objective function component takes the form

$$\begin{aligned} E_{\mathrm{spt},g}(X) &= \frac{1}{2}\,||\kappa_p(g_{\mathrm{ref}} - g) - \kappa_v\dot{g} - \ddot{g}||^2\,, \\ &= \frac{1}{2}\,X^T Q X + c^T X + \frac{1}{2}\,c^T c\,, \end{aligned} \tag{13}$$

where

$$Q = \begin{pmatrix} J_g^T J_g & 0_{\alpha\times\beta} & 0_{\alpha\times\gamma} \\ 0_{\beta\times\alpha} & 0_{\beta\times\beta} & 0_{\beta\times\gamma} \\ 0_{\gamma\times\alpha} & 0_{\gamma\times\beta} & 0_{\gamma\times\gamma} \end{pmatrix}, c = \begin{pmatrix} -J_g^T\left(\kappa_p(g_{\mathrm{ref}} - g) - \kappa_v J_g\dot{q} - \dot{J}_g\dot{q}\right) \\ 0_\beta \\ 0_\gamma \end{pmatrix} \tag{14}$$

. κ_p and κ_v are hand-tuned gain parameters, in our applications we systematically set $\kappa_v = 2\sqrt{\kappa_p}$.

The Target Objective: Let t_0 denote the current time. Let g^i be the i-th scalar component of g for $i \in \{1, \ldots, d\}$. For every such g^i our objective is to reach the specified target $(g_f^i, \dot{g}_f^i)$ at time $t_f > t_0$. The method proposed in [8] consists in making g^i follow a constant-jerk reference trajectory of the form

$$\ddot{g}^i_{\mathrm{ref}}(t) = \left(1 - \frac{t - t_0}{t_f - t_0}\right)\,\phi_{i,t_0} + \frac{t - t_0}{t_f - t_0}\,\psi_{i,t_0},\ t \in [t_0, t_f] \tag{15}$$

where ϕ_{i,t_0} and ψ_{i,t_0} are coefficients determined by integrating (15) twice and writing the boundary values conditions

$$\begin{pmatrix} (t_f - t_0)^2/3 & (t_f - t_0)^2/6 \\ (t_f - t_0)/2 & (t_f - t_0)/2 \end{pmatrix}\begin{pmatrix} \phi_{i,t_0} \\ \psi_{i,t_0} \end{pmatrix} = \begin{pmatrix} g_f^i - g^i - (t_f - t_0)\,\dot{g}^i \\ \dot{g}_f^i - \dot{g}^i \end{pmatrix}\,. \tag{16}$$

Finally, back to the target objective, the corresponding objective function component will take the form

$$\begin{aligned} E_{\mathrm{tgt},g}(X) &= \sum_{i=1}^{d}\frac{1}{2}\,(\ddot{g}^i_{\mathrm{ref}}(t_0) - \ddot{g}^i)^2 = \sum_{i=1}^{d}\frac{1}{2}\,(\phi_{i,t_0} - \ddot{g}^i)^2\,, \\ &= \frac{1}{2}\,X^T Q X + c^T X + \frac{1}{2}\,c^T c\,, \end{aligned} \tag{17}$$

where, denoting $\Phi_{t_0} = (\phi_{i,t_0})_i$,

$$Q = \begin{pmatrix} J_g^T J_g & 0_{\alpha\times\beta} & 0_{\alpha\times\gamma} \\ 0_{\beta\times\alpha} & 0_{\beta\times\beta} & 0_{\beta\times\gamma} \\ 0_{\gamma\times\alpha} & 0_{\gamma\times\beta} & 0_{\gamma\times\gamma} \end{pmatrix}, c = \begin{pmatrix} -J_g^T\left(\Phi_{t_0} - \dot{J}_g\,\dot{q}\right) \\ 0_\beta \\ 0_\gamma \end{pmatrix}\,. \tag{18}$$

C. Putting it Altogether: The QP Formulation

We suppose now that we have N objectives indexed by $k \in \{1, \ldots, N\}$, denoted $g_1, \ldots, g_N$. These objectives can be either set-point or target objectives, with corresponding matrices Q_k and vectors c_k as derived in the previous section. Each objective g_k is allocated a weight w_k that expresses its relative importance when conflicting with other objectives. We then denote the weighted sums

$$Q_{\mathrm{sum}} = \sum_{k=1}^{N} w_k\, Q_k\,, \quad c_{\mathrm{sum}} = \sum_{k=1}^{N} w_k\, c_k\,. \tag{19}$$

The Quadratic Program solved by the multi-objective controller at every time step takes the final form

$$\begin{aligned} \min_X \quad & \frac{1}{2}\,X^T Q_{\mathrm{sum}} X + c_{\mathrm{sum}}^T X\,, \\ \text{subject to} \quad & A_1\, X = B_1\,,\ A_2\, X \leq B_2\,. \end{aligned} \tag{20}$$

V. FINITE-STATE MACHINE

Let us us now start back from the output of the multi-contact stances planner as portrayed in Fig. 1, which is a sequence of n statically stable configurations $(q_0, \dots, q_{n-1})$. Each configuration q_i is associated with a *stance* σ_i; a stance being the set of contacts that the robot establishes with the environment when put in that configuration. For example when the robot stands on two feet then the corresponding stance is a set containing two contacts (one for each foot). The sequence of stances $(\sigma_0, \dots, \sigma_{n-1})$ output by the planner are so-called *sequentially adjacent* [2], i.e. they satisfy the following condition: each stance σ_i either adds one contact to the previous stance σ_{i-1} or removes one contact from this same previous stance σ_{i-1}. Furthermore, the sequence of configurations $(q_0, \dots, q_{n-1})$ are so-called *transition configurations* [2], meaning that:

- when a contact has been added then the corresponding configuration q_i has to be statically stable with non-zero contact forces applied only at the contacts of the previous stance σ_{i-1}, the contact forces applied at the newly added contact are zero,
- when a contact has been removed then the corresponding configuration q_i keeps all the contacts of the previous stance σ_{i-1} but the contact forces at the newly removed contact are zero.

The motion from q_i to q_{i+1} (from σ_i to σ_{i+1}) will be called *step* number i. So the full motion will comprise $n-1$ steps. We define a user-input parameter T which is the desired step time. So step i starts at time $t = i\,T$ and ends at time $t = (i+1)\,T$. The full duration of the motion is $(n-1)\,T$.

When step i adds a contact then the link of the added contact (the "swing" link, generalizing the terminology of swing foot in legged locomotion) will be denoted s_i and one arbitrarily chosen point attached to this link and belonging to the contact surface is denoted p_i. The global-frame-expressed position of p_i at configuration q_i (start position) is denoted $P_{i,s}$ and the global-frame-expressed position of p_i at configuration q_{i+1} (goal position) is denoted $P_{i,g}$.

A. Obstacle Collision Avoidance: Controlling the Swing Link

When step i is removing a contact we implicitly make the assumption that the motion from q_i to q_{i+1} (performed inside the sub-manifold of the configuration space corresponding to the stance σ_i) is collision-free. When step i adds a contact however, then the motion of the swing link s_i has to be more carefully controlled since there is high probability that this link collides with the target environment contact support object; e.g. when climbing stairs then the swing foot might collide with the next stair. We introduce a simple heuristic to avoid this, which consists in steering the swing point p_i through a global-frame-expressed *via-point* $P_{i,v}$, defined by specifying a step height h and an intermediate time $T_v < T$ (for example one might choose $T_v = T/2$). So the motion of p_i starts from $P_{i,s}$ at time $t = i\,T$, goes through $P_{i,v}$ at time $t = i\,T + T_v$, and reaches $P_{i,g}$ at time $t = (i+1)\,T$.

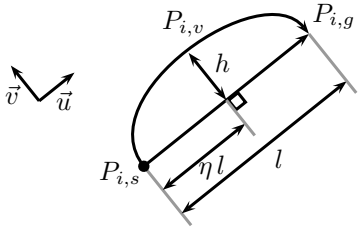


Fig. 2. Controlling the point p_i of the swing link s_i

Let us denote the step length $l = \|P_{i,g} - P_{i,v}\|$. To define the via-point $P_{i,v}$ we decompose the motion of the swing point p_i into a *parallel component* in the direction of the vector $\vec{u} = (P_{i,g} - P_{i,v})/l$, and a *normal component* following the direction of the vector $\vec{v} = \vec{u} \times (\vec{e}_z \times \vec{u})$ (such that $\vec{v}$ is normal to $\vec{u}$ and in the plan defined by $\vec{u}$ and $\vec{e}_z$; $\vec{e}_z$ being the upwards vertical unit vector opposite to the gravity). The via-point is finally defined as

$$P_{i,v} = P_{i,g} + \eta\,l\,\vec{u} + h\,\vec{v}, \quad \eta \in [0,1]\,. \tag{21}$$

A typical choice of the parameter η is $\eta = 1/2$. See Fig. 2.

Furthermore, we impose that the swing point p_i reaches its goal $P_{i,g}$ at time $t = (i+1)\,T$ with zero velocity, and that it reaches its via-point $P_{i,v}$ at time $t = i\,T + T_v$ with a zero $\vec{v}$-component (normal) velocity.

All these objectives are formulated as *target objectives*.

B. Keeping Balance: Controlling the CoM

The balance of the simulated robot is controlled through simple strategies, depending on whether we are adding or removing a contact. If step i adds a contact from then, following the *transition configurations* condition, the whole motion has to be performed by staying balanced on the initial stance σ_i, so the objective for the CoM in this case is a *set-point objective* that regulates its position around its position at the start configuration q_i. If step i removes a contact, then the robot has to "transfer its weight" from stance σ_i to stance σ_{i+1} in time T. For this purpose a *target objective* is defined for the CoM to reach at time $(i+1)\,T$ its position computed at the goal configuration q_{i+1}, with zero velocity.

C. Solving the Redundancy: Controlling the Configuration

The remaining redundancies are solved by controlling the whole configuration of the robot, with once again different strategies when adding or removing a contact. When adding a contact at step i, the posture is controlled through a set-point objective with the reference posture being set at $q_{\text{ref}} = q_i$ for the time interval $t \in [i\,T, i\,T + T_v]$ and set at $q_{\text{ref}} = q_{i+1}$ for the time interval $t \in [i\,T + T_v, (i+1)\,T]$ with low stiffness κ_p. When removing a contact then the reference configuration for the low-stiffness set-point objective is set at $q_{\text{ref}} = q_{i+1}$ during the whole step time interval $t \in [i\,T, (i+1)\,T]$.

D. Putting it Altogether: the FSM

As a summary of this section, Fig. 3 shows a graphical representation of the FSM. Details of the objectives are found in the previous subsections. The initial configuration of the robot at time $t = 0$ is $q = q_0 = q_s$ with $\dot{q} = 0$.

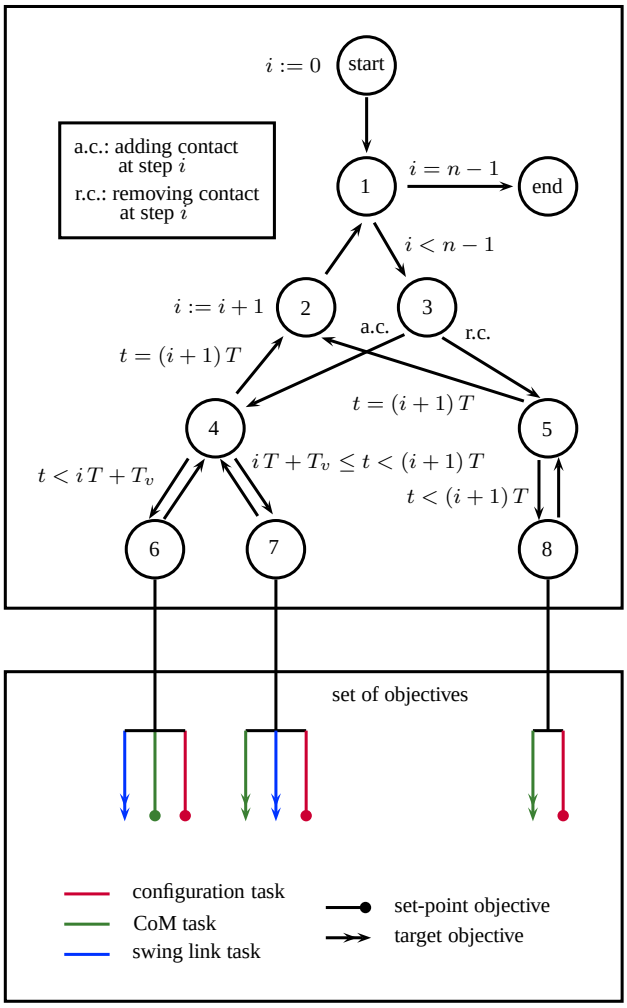


Fig. 3. The finite-state machine (note: contains color information). States are represented as circles (the numbers inside have no particular meaning) and transitions as arrows between states. Labels next to transitions are the conditions for the transitions to be triggered. Transitions without labels are automatically triggered (condition always true). Labels next to states, when present, are actions performed when the machine reaches the states.

VI. Playback Simulation Results

The video attached to this paper shows some example applications of the proposed approach. These examples are: a basic walk motion, a single stair climbing motion, a multiple stairs climbing motion, a sitting motion, a one-step walk-on-hands motion. See Fig. 4 for snapshots of this video.

A. Experimental Framework

The humanoid robot model used is HRP-2 [16] with some modifications in terms of torque limits and arm links for the walk-on-hand motion, though our implementation is transparent to the particular robot model. The simulator used is described in [14] and the multi-contact static stances planner in [2]. Collision detection between the robot and the environment is performed using the PQP proximity queries package [17], and the QP solver used for multi-objective control is the QL convex quadratic programming solver [18]. Table I gives the parameters used for these motions.

The video starts by showing elementary motions (single steps) produced by the multi-objective controller with fixed objectives. Then the five above-mentioned motions generated by coupling the multi-objective controller with the finite-state machine are sequentially played. Each of the five motions starts by first showing the output of the multi-contact stances planner used as input for that motion, i.e. the finite sequence of static postures $(q_0, \ldots, q_{n-1})$.

B. Discussion and Limitations

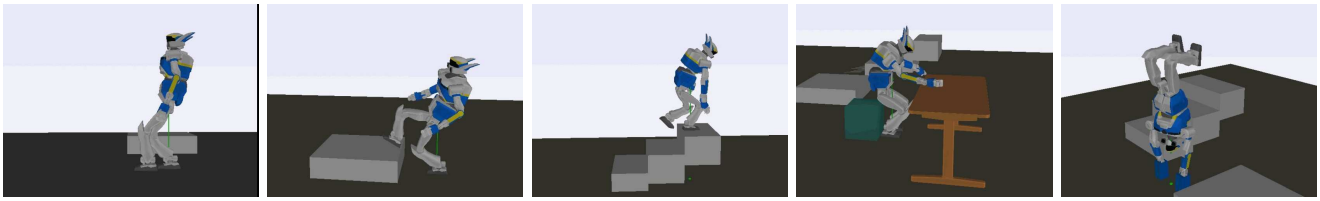
The motions displayed on this video have not been generated in real time. We used a time step of 1 ms for the simulator, but each iteration of the motion generator cycle took approximately 30 ms to compute on our 3 GHz Pentium IV system. However, real-time on-line control is not, at this stage, the main preoccupation of our work, so no particular effort has been devoted to reducing this computation time in our prototype implementation. Still, as a motion generation tool, the method is much faster than global motion optimization techniques [13].

Another limitation that currently prevents our method from being used as such as a control tool for the real robot is the absence of self-collision checking in the simulation (walk scenario), and joint limits constraints (single stair scenario). A basic strategy to reduce self-collision occurrences and to stay within joint limits that we implemented is the introduction of repulsive torques that are activated when a joint comes too close to its limit, but this does not absolutely guarantee that the limits are not reached.

An interesting feature that appears in these motions is the robustness to collisions with the environment and to uncertainty with regard to contact locations. In particular, we can see that when climbing the stairs, the swing foot can slightly collide with the stair but the robot does not lose balance. Also, even if the contact is not precisely put at its planned position this does not prevent the motion from being successfully carried out on the stance including that contact. These remarks are encouraging in the perspective of later using the method for the control of the real robot.

There were cases however in which the collision of the swing link with the environment led to an impact from which the robot could not recover and ended up falling down. A posteriori tuning of the CoM objectives weights and gains sometimes enabled to regenerate a stable motion.

"Falling down" is what happens when the constraints of the QP (19) cannot be satisfied. This means that the robot reached a state $(q, \dot{q})$ outside of the *viability kernel* [19]. If we had used a prioritization approach, this would have led to either a dynamically feasible motion that breaks the contacts, or a non-dynamically-consistent motion that maintains the contacts, both cases resulting in an ill-posed QP formulation in the subsequent simulation step. This is why we did not see the necessity to use prioritized formulation, and the motion generation fails ("crashes") in case the robot reaches such a non-viable state. Recovery strategies from these situations should be further investigated. Note however that falling down can also occur while all the constraints are satisfied,



(a) basic walk (b) single stair climbing (c) multiple stairs climbing (d) sitting (e) one step on hands

Fig. 4. Snapshots from the attached video

TABLE I

MOTION GENERATION PARAMETERS

	walk	single stair	multiple stairs	sitting	hands		all scenarios
number of steps n	10	6	8	3	2	w_{config}	10^1
step duration T	0.8 s	1.5 s	4 s	4 s	4 s	w_{CoM}	10^4
step height h	1 cm	30 cm/10 cm	55 cm/30 cm	10 cm/0 cm	10 cm	w_{slink}	10^3
parameter T_v	0.4 s	0.75 s	2 s	2 s	2 s	$\kappa_{p,\text{config}}$	10^1
parameter η	0.5					$\kappa_{p,\text{CoM}}$	10^3

since the CoM control strategy does not strictly quantify the "stability keeping" notion that is not well defined outside a ZMP-applicable framework (e.g. [19] for a discussion).

VII. Conclusion and Future Work

We investigated a method inspired from computer graphics animation to generate simulated humanoid robot motion. This method allows the robot to benefit from full autonomy from the multi-contact planning stage to the motion generation stage, and thus the two stages of the contact-before-motion framework are achieved.

A number of issues have to be addressed to convert this motion generation tool into an on-line control tool. Most important is reaching real-time performance. Collision avoidance constraints might be included in the QP formulation by using repulsive potential field approaches.

We are also studying extendibility to problems such as object manipulation and multiple robot collaboration, as our generic multi-contact stances planner can handle these.

Another possible improvement worth investigating is to add to the framework a reduced-model planning phase that would produce more dynamic motions.

Solving these issues would make us a step closer to the longer-term pursued objective of real-time full autonomy of humanoid robots.

Acknowledgement

This work is partially supported by Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (B), 22300071, 2010.

References

[1] K. Bouyarmane and A. Kheddar, "Static multi-contact inverse problem for multiple humanoid robots and manipulated objects," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 2010.

[2] ——, "Multi-contact planning for multiple agents," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2011.

[3] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on HRP-2," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[4] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 2005.

[5] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, pp. 105–118, 2002.

[6] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond, "Humanoid motion planning for dynamic tasks," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 2005.

[7] Y. Abe, M. da Silva, and J. Popovic, "Multiobjective control with frictional contacts," in *Eurographics/ACM SIGGRAPH Symp. on Computer Animation*, 2007.

[8] M. de Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 29, no. 3, 2010.

[9] K. Yamane and J. Hodgins, "Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 2010.

[10] J. Salini, S. Barthelemy, and P. Bidaud, "LQP controller design for generic whole body motion," in *Climbing and Walking Robots and the Support Technologies for Mobile Machines*, 2009.

[11] C. Collette, "Virtual humans dynamic control : robust balance and task management," Ph.D. dissertation, University of Paris VI, June 2009.

[12] L. Sentis, J. Park, and O. Khatib, "Compliant control of multi-contact and center of mass behaviors in humanoid robots," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 483–501, 2010.

[13] S. Lengagne, P. Mathieu, A. Kheddar, and E. Yoshida, "Generation of dynamic multi-contact motions: 2d case studies," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 2010.

[14] J.-R. Chardonnet, S. Miossec, A. Kheddar, H. Arisumi, H. Hirukawa, F. Pierrot, and K. Yokoi, "Dynamic simulator for humanoids using constraint-based method with static friction," in *Proc. of the IEEE Int. Conf. on Robotics and Biomimetics*, 2006.

[15] P.-B. Wieber, "Some comments on the structure of the dynamics of articulated motion," in *Fast Motions in Biomechanics and Robotics*, 2005.

[16] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2004.

[17] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2000.

[18] K. Schittkowski, "QL: A fortran code for convex quadratic programming - user's guide," Department of Computer Science, University of Bayreuth, Tech. Rep., 2007.

[19] P.-B. Wieber, "On the stability of walking systems," in *Humanoid and Human Friendly Robotics*, 2002.