2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
September 14-18, 2014. Chicago, IL, USA,

Model Preview Control in Multi-Contact Motion– Application to a Humanoid Robot

Hervé Audren[2,1], Joris Vaillant[2], Abderrahmane Kheddar[1,2], Adrien Escande[1], Kenji Kaneko[1], Eiichi Yoshida[1]

***Abstract*— Our work builds largely on Nagasaka's stabilizer in multi-contact motion [1]. Using a sequence of contact stances from an offline multi-contact planner, we use first a Model Predictive Controller to generate a dynamic trajectory of the center of mass, then a whole-body closed-loop model-based controller to track it at best. Relatively to Nagasaka's work, we allow frame changes of the preferred force, provide a heuristic to compute the timing of the transition from purely geometrical features and investigate the synchronization problem between the reduced-model preview control and the whole-body controller. Using our framework, we generate a wide range of 3D motions, while accounting for predictable external forces, which includes transporting objects. Simulation scenarios are presented and obtained results are analyzed and discussed.**

I. Introduction

Multi-contact motion planning and closed-loop control of humanoid robots is useful in many situations. Indeed, having humanoids capable of moving by taking additional contacts (when needed) that make use of any of their links, allows them to evolve in cumbersome environments and to enforce their equilibrium, preventing them from falling. Multi-contact dynamic motion is sometimes wrongly understood or misused. Indeed, *dynamic* does not only mean computing motion using the dynamic models, it requires also the ability to forecast at least one or two contact sequences ahead and exploit robots dynamic to generate the motion that will go through. How much of the contact sequence need to be known ahead depends a lot on the tasks and its conditions.

There are diverse multi-contact control strategies. Strictly prioritized task-space controllers that compute motions using the dynamic model of the robot are used in [2], [3]. Weighted prioritized task-space controllers are proposed in [4], [5], [6], [7], [8]. These controllers proved to be efficient, although their robustness relies heavily on the numerical solver they use (generally off-the-shelf or customized [9] QP solvers). Yet, they do not anticipate upcoming tasks, e.g. the next contact sequences, and they require the user to set the timing of the tasks. Whole-body model preview control (MPC) that exploits dynamics and future contact sequence can only be formulated as a nonlinear optimal control or program as in [10]. In this case, if the computation of such multi-contact trajectories can be made efficiently within a second or so, it can be used in a closed-loop MPC scheme. We are far from it: this is the reason why early dynamic walking approaches use a two-level scheme:

1) compute quickly the dynamic motion (with a time horizon window taking into account few next steps) by means of a simplified dynamic model that yet can capture the essence of the whole body dynamics, e.g. the center-of-mass (CoM);
2) provide the CoM computed trajectory to be tracked at best by whatever chosen local low-level whole body motion planner/controller.

One problem with such a scheme is that there is no guarantee that the tracking of the CoM by the second level is always feasible. This is why heuristics and hypotheses specific to tasks to be achieved are used. Yet, this approach is successfully used in humanoid biped walking illustrated by the seminal work of Kajita et al. [11] that uses a model preview control (MPC) of the ZMP. More impressive enhancements were illustrated in computer graphics [12] with a spring-load inverted pendulum (SLIP).

CoM-based models appear, until recently, to be difficult to extend beyond walking. However, Nagasaka et al. [1] proposed an elegant formulation of MPC using the CoM for multi-contact. The present contribution is the outcome from our attempt to implement [1] in the framework of our multi-contact planner [13], [14]. We report the problems that we encountered and the practical enhancements made. Besides this, and relative to Nagasaka's method, our additional contributions are as follows: (i) the possibility to choose any desired preferred axis (change of referential), (ii) handling external sustained forces; (iii) automatic determination of timings from spatial information; (iv) a new position/velocity CoM control instead of force (an equality constraint on contact forces was used in [1]); (v) application to a combined locomotion/manipulation.

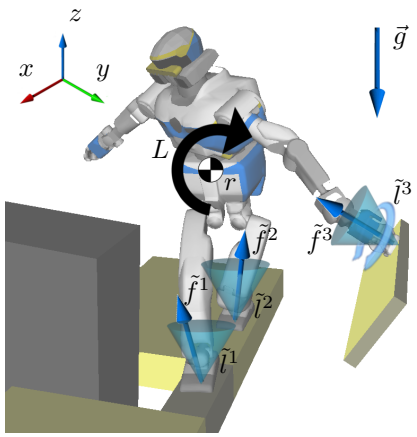

Fig. 1. HRP-2 leaning: representation of variables used in modeling.

[1]CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/CRT, Japan
[2]CNRS-UM2 LIRMM, Interactive Digital Humans (IDH), France

II. Model Predictive Control

A. Previous Formalism

In [1], the robot's flywheel model state equation is derived from the conservation of the linear and angular momentum, projected on x and y axes, see Figure 1:

$$m\ddot{r} = \sum_{i=1}^{n} f^i - mg \tag{1}$$

$$\dot{L} = \sum_{i=1}^{n} [(p^i - r) \times f^i + l^i] \tag{2}$$

with f^i and l^i the force and momentum applied at i-th (contact) point p^i, m the total mass of the robot, r the position of the CoM and L the angular momentum at the CoM across the number of contacts n. Upon using a discrete integration scheme, this choice yields a linear formulation as long as L_z is ignored and the sum of external forces on the z-axis is constant and known at each discrete time interval $[t_k, t_{k+1}]$; that is, we provide F_{z_k} and impose

$$F_{z_k} = \sum_{i=0}^{n_k} f^i_{z_k} \tag{3}$$

n_k being the number of contacts at instant k. Then, we build a state vector composed of: the linear momentum, their integral and the angular momentum all of which are along the x- and y-axes. In [1], the z-axis is aligned with the gravity $-\vec{g}$. Recall that we have uncontrolled angular momentum around the z-axis since $\dot{L}_z$ is non-linear. The command vector is composed of the forces and momenta applied at each centroid of the contact polygons. Under the constraint (3) and the assumption that the centroid positions of the contact-polygons are known, we have:

$$\hat{x}_{k+1} = A_k \hat{x}_k + B_k u_k \tag{4}$$

$$\hat{x}_k = \begin{bmatrix} mr_x & m\dot{r}_x & mr_y & m\dot{r}_y & L_x & L_y \end{bmatrix} \tag{5}$$

$$u_k = \begin{bmatrix} f^1_x & f^1_y & f^1_z & l^1_x & l^1_y & l^1_z & \dots & l^{n_k}_z \end{bmatrix} \tag{6}$$

Let $X = \begin{bmatrix} \hat{x}_1^T & \dots & \hat{x}_K^T \end{bmatrix}^T$ and $U = \begin{bmatrix} u_0^T & \dots & u_{K-1}^T \end{bmatrix}^T$, we can apply (4) repeatedly over a time-horizon window $[t_0, t_K]$ to get

$$X = \Phi \hat{x}_0 + \Psi U \tag{7}$$

$\hat{x}_0$ is the given initial state. We define a quadratic cost in contact wrenches U to be minimized while tracking a given reference state trajectory $\overline{X}$ (explained later in section III-C):

$$\begin{aligned} V &= (X - \overline{X})^T W_x (X - \overline{X}) + U^T W_u U \\ &= U^T Q U + 2U^T v + \gamma \quad \text{(by using eq. (7))} \\ Q &= \Psi^T W_x \Psi + W_u \\ v &= \Psi^T W_x \Phi \hat{x}_0 - \Psi^T W_x \overline{X} \end{aligned} \tag{8}$$

Additionally, we impose constraints on forces to sustain contacts (that are unilateral) and non-sliding using the associated (4-sided) linearized friction cones written in each contact reference frame (here, $\tilde{z}$ is the contact normal direction). Assuming that the local-to-world frame transforms are known, these constraints write:

$$\tilde{f}^i_{k_z} \geq 0 \ \ \forall i \in [1, n_k] \, \forall k \in [0, K-1] \tag{9}$$

$$|\tilde{f}_{k_{x|y}}| \leq \mu^i_k f_{k_z} \ \ \forall i \in [1, n_k] \, \forall k \in [0, K-1] \tag{10}$$

We also apply a center of pressure (CoP) condition on each line (a, b, c) of each contact polygon's edge at each time k:

$$\begin{aligned} -a^i_{k_j} \tilde{l}^i_{k_y} + b^i_{k_j} \tilde{l}^i_{k_x} + c^i_{k_j} \tilde{f}^i_{k_z} \geq 0 \\ \forall k \in [0, K-1] \, \forall i \in [1, n_k] \, \forall j \in [1, m_k] \end{aligned} \tag{11}$$

m_k is the total number of all contact's polygon edges. Minimizing (8) under constraints (3) and (9)–(11) is a QP which solution gives us the optimal set of contact forces (in their local frames) that will be substituted in (7) from which a feasible X is computed to be tracked at best by the lower level controller. W_x and W_u are QP diagonal tuning gains.

B. Arbitrary reference frames

In [1]'s formulation the z-axis is aligned with $-\vec{g}$. This is limiting in cases where general direction of motion is along the gravity field (e.g. climbing a ladder or spider-walking): setting the z force trajectory is rather limiting and can even be not feasible. One would instead set the transversal swaying to zero rather than the 'climbing' trajectory which can preferably be left to the planning process.

Now, we allow choosing an arbitrary direction along which one component of the force trajectory is set. Therefore, we will account for the gravity components and project eqs. (1) and (2) onto an arbitrary reference frame $\mathcal{W}\{\vec{w}_1, \vec{w}_2, \vec{w}_3\}$. We can then rewrite the state vector and the state equation:

$$\hat{x}_{k+1} = A_k \hat{x}_k + B_k u_k + C_k \tag{12}$$

$$\hat{x}_k = \begin{bmatrix} mr_1 & m\dot{r}_1 & mr_2 & m\dot{r}_2 & L_1 & L_2 \end{bmatrix} \tag{13}$$

where C_k represents the contribution of the gravity in $\mathcal{W}$:

$$C_k = \begin{bmatrix} -\frac{T_k^2}{2} \mathcal{R}_{k,1} \overrightarrow{g} \\ -T_k \mathcal{R}_{k,1} \overrightarrow{g} \\ -\frac{T_k^2}{2} \mathcal{R}_{k,2} \overrightarrow{g} \\ -T_k \mathcal{R}_{k,2} \overrightarrow{g} \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

This leads us to a new formulation of eq. (7):

$$X = \Phi \hat{x}_0 + \Psi U + \begin{bmatrix} C_0 \\ A_1 C_0 + C_1 \\ A_2 A_1 C_0 + A_2 C_1 + C_2 \\ \vdots \\ \sum_{i=0}^{K} \left(\left[\prod_{j=i+1}^{K} A_j \right] C_i \right) \end{bmatrix} \tag{15}$$

As the command vector U is written in local reference frames, we do not need to change neither the constraints nor the basic expression of our matrix Ψ, which now depends on the transform matrix from local frames to $\mathcal{W}$. Note

that problems may occur if the privileged direction changes within the preview window. This problem will be thoroughly investigated in future work.

C. Dealing with other external known forces

For the time being C_k is used to represent the effect of gravity in $\mathcal{W}$. Yet, this term can actually encompass any other external known force. For example, we can account for sustained forces due to holding an object during motion. But, for the preview to compensate accurately for this additional force, it is necessary to know at what time the force will apply and how much force and momentum is applied at the CoM (hence, where it will be applied). This is not possible in general because the force will be applied at a point that is moving and depend on the actual position of the robot which we do not know a priori.

III. The objective-based controller

A. Presentation

Our multi-contact controller is composed of the modules illustrated in Figure 2. First, a multi-contact planner outputs a sequence of stances [13]. Each stance is composed of a set of contacts and a statically-stable configuration of the robot (including the free-flyer). A QP task-based controller, described in [15], [16] accounts for various types of constraints and tasks (such as CoM tracking) to achieve multi-contact stance-to-stance transitions. This position command is then sent to the robot's PD control loop.

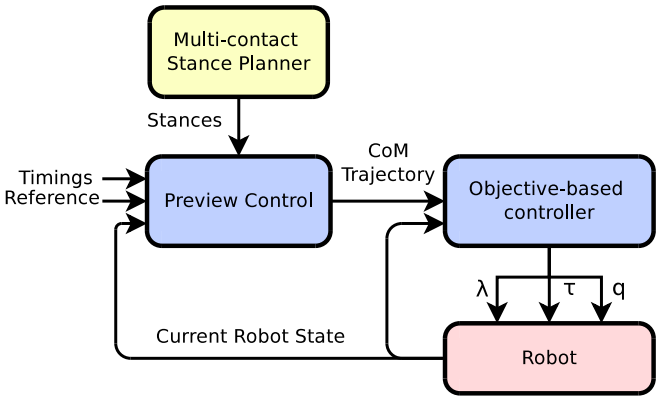

Fig. 2. Multi-contact motion planning architecture.

Our preview control in Figure 2 requires two parameters:

1) setting the timings i.e. the instants at which the contacts are either made or released, and
2) a reference, noted $\overline{X}$ in subsection II-A.

B. Time parametrization of the stances

The other price to pay for having a linear formulation of the problem is to predefine the timings of the stances. Indeed, our multi-contact planner does not produce any temporal information whereas our preview controller requires it. We tried setting intuitive ad-hoc values; but it quickly turned out to be a very critical issue as many simulated scenarios failed if the robot was given too much or not enough time to complete desired multi-contact motions. Nonlinear optimization techniques (resolving for the timing) are still computationally costly and not very robust [10]. Instead, we investigated whether using heuristics from spatial information (provided by the multi-contact planner) can be an acceptable substitute. To do so, we categorized the transitions between stances into *support*, when the CoM moves with sustained contacts, and *move*, when the number of contacts changes. For the *support* transitions, it appears that the distance between the CoM of the stances is a good metric for the time needed to accomplish the transition. For the *move* stances, we need the distance travelled by the link currently changing its contact state. We chose a cubic polynomial that links the start-point $\vec{p}_0$ and the end-point $\vec{p}_1$ (given by the stance configuration), arriving with zero velocity, plus a waypoint defined as follows:

$$\vec{p}_{wp} = \vec{p}_0 + d_{\parallel}(\vec{p}_1 - \vec{p}_0) + d_{\perp}(\vec{p}_1 - \vec{p}_0)^{\perp} \qquad (16)$$

Typical values we use are: $d_{\parallel} = 0.1$, $d_{\perp} = 0.15$. We define $\vec{v}^{\perp} = \frac{\vec{n}}{\|\vec{n}\|} \mid \vec{n} \in \text{plane}\,(\vec{v}, \vec{z})$ and $\vec{n} \cdot \vec{z} \geq 0$ and $\vec{n} \cdot \vec{v} = 0$.

All our simulations did not need anything more complex than this heuristic to behave well. These trajectories need to be recomputed online as the starting point depends on the actual position of the robot. Techniques such as parallel parameter space exploration [17] or CHOMP [18] can be used to generate more reliable trajectories than cubic polynomials. We propose the following heuristic for timing:

$$t_{\nu+1} = t_{\nu} + \frac{\{d, l\}}{v_0}\left(1 - \left(\frac{\min(t_n, \tau)}{\alpha}\right)^{\kappa}\right) \qquad (17)$$

meaning that for transiting from stance ν to $\nu + 1$, the CoM motion starts with a desired given speed v_0 and accelerates in a degree κ during a time τ. The time scale α regulates the final speed for each stance. In addition, $\alpha > \tau$ to keep positive timings. d or l are respectively the distance between $\text{CoM}_{\nu+1}$ and CoM_{ν}, and the length of contacting point computed trajectory. This heuristic describes a strategy similar to a trapezoidal speed command but with no deceleration. Simulations show that adding deceleration did not improve much the behavior. After careful tuning of the heuristic for our robot, we set: $v_{0,d} = 0.75$, $v_{0,l} = 0.5$, $\kappa = 2$, $\tau = 4$ and $\alpha^2 = 50$.

C. Defining the reference $\overline{X}$ and tuning the MPC gains

Defining a desired reference $\overline{X}$ for our MPC also amounts to tuning the weight matrices W_x and W_u.

The desired angular moments are easy: as we do not want the robot to sway, we will impose on them zero reference with high weights, $\overline{L}_x = 0, \overline{L}_y = 0$ in $\overline{X}$. For the remaining $m\overline{r}_x, m\overline{r}_y$ and their derivatives, the reference is given as a constant speed interpolated between the first and the last stance. Associated weights W_x are put higher on the speed relatively to position. This simple setting allows fast computations on-the-fly. Moreover, this speed reference is continuous, whereas piecewise interpolation between stances gave worse results because of abrupt changes in acceleration.

Yet, interpolation would not be applicable in strongly curved paths where the robot has to change motion direction. Subsequently, we approximate the path by a sequence of gross segments to better match it.

As for the setting of the privileged axis ($\vec{u}_3$), we chose it simply as a trajectory at constant velocity, obtained from the ratio of the distance separating final from current CoMs along $\vec{u}_3$ divided by the difference between final and current times.

Concerning W_u tuning, we want the robot to follow the trajectory first, and then minimize the contact forces, we will put much lower weights w.r.t. W_x.

IV. Lifting an object

Our MPC accounts for a given external force. A good example to assess the latter claim is to lift an object and walk with it. We discuss three approaches we considered:

On way to deal with an additional hold object is to integrate contacts' locations and forces between the robot and the objet as part of the MPC with additional constraints on grasp stability. This however increases the complexity of the problem (that will end as a non-linear formulation); therefore, we do not use it.

Apart from using robot/object contact forces (left for the QP task controller), the second way could be to exploit the external forces term C_k in subsection II-B. Indeed, C_k can also embed the weight of the object: we simply need to compute the weight and moment produced by the object w.r.t the CoM. By doing so, we are able to account, in the MPC, for an " instant" pick-up when the robot lifts the object. Unfortunately, it will be hard to predict exactly the object's trajectory ahead of time because this is left to –and results from– the task controller. It is then difficult to predict the moment applied by the object w.r.t the CoM along the preview window.

Third, we consider our reduced point-mass system to change from the *robot* system to the *robot + object* system. This turns the constant mass m into two values of the masses: m_{robot} or $m_{\text{robot + object}}$ at each sample k. By doing so, the CoM position and velocity will be discontinuous at the pick-up phase. This discontinuity is 'filtered' by the lower-level task-based controller when possible. We use this approach and assess it with simulations.

A. Change in the QP task controller

We modify the task-based QP controller to include the dynamics of the manipulated object. We extend the controller state with the object's state because it is the best way to have the robot use the object to regulate its own dynamics, yielding a much plausible plan that eventually allows performing real experiments. However, this solution requires identifying the inertia parameters (we may not know the precise weight of the manipulated body). Instead, we rather impose a force task on both wrists to maintain the manipulated object in position. We also consider a different kind of contacts between the robot and the manipulated body, thus changing the state vector into:

$$x = \begin{bmatrix} \ddot{q}_{\text{robot}} & \ddot{q}_{\text{object}} & \lambda_{e\to r} & \lambda_{o\leftrightarrow r} & \tau \end{bmatrix} \tag{18}$$

subscript e stands for environment, o for manipulated object and r for robot. The arrow denotes a contact between two entities, applied on the latter. Note that the reciprocal pair of contacts between the robot and the manipulated body use the same forces intensity, λ on the same friction cone generators, so that the forces are effectively reciprocal. This new formulation leads us to design two new equality constraints, written as $Ax = B$ for the extended dynamic motion constraint, that will allow us to compute the object's acceleration $\ddot{q}_o$. We have:

$$A = \begin{bmatrix} H_r & 0 & \lceil_{i\in e\to r} J_i\mu_i\rceil & \lceil_{i\in o\to r} J_i\mu_i\rceil & \begin{bmatrix}0_6\\ -I\end{bmatrix} \\ 0 & H_o & 0 & \lceil_{i\in r\to o} J_i\mu_i\rceil & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} C_{\text{robot}} \\ C_{\text{object}} \end{bmatrix} \tag{19}$$

where H is the inertia matrix, J is the Jacobian, μ is the friction cone generator, C represents the nonlinear gravitational effects, subscribes o, r and e are defined as previously. The $\lceil\ \rceil$ operator denotes the horizontal concatenation of the enclosed quantity across all generators of each contact. We also add a manipulation acceleration constraint such that for every contact between the robot and the object, at each contact point:

$$J_{o\to r}\ddot{q} + \dot{J}_{o\to r}\dot{q} = J_{r\to o}\ddot{q} + \dot{J}_{r\to o}\dot{q} \tag{20}$$

Which translates, in our implementation into:

$$A = \begin{bmatrix} -J_{r\to o} & J_{o\to r} & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} \dot{J}_{r\to o}\dot{q} - \dot{J}_{o\to r}\dot{q} \end{bmatrix} \tag{21}$$

Finally the remaining changes are as follows:

First, we enhanced the initial Finite State Machine of [15] to account for additional steps that are: (i) reach a target posture before lifting an object, (ii) trigger on/off the necessary modifications to bilaterally switch between *robot* and *robot+object* (iii) bridge on/off to the robot multi-contact FSM.

For CoM and angular momentum tracking tasks, we compute these quantities by assuming the ensemble robot + object attached – recall that the MPC outputs results considering the total mass and the resultant CoM for the reduced model. This assumption is plausible since enforced by non-slip constraints that we impose on the contact points between the end effectors and the object.

To sum-up, our controller computes torques and joint accelerations for the robot and manipulated body (see [15]):

- The equality and inequality constraints are:
  - Dynamics of the object and robot (19);
  - Non-sliding contacts between the robot and the environment and the object (21);
  - Torque and joints limits;
  - Collision avoidance with itself and the environment.

- The objective function composed of:
  - CoM objective: position, velocity and angular momentum track the output of the preview control;
  - Posture objective: match at best the stances generated by the static multi-contact planner;
  - Contact objectives: activate on/off target contact orientation and position tasks.

V. Results

All the results presented in this section are illustrated by the attached video, including output of the stance planner and actual motion.

A. Walking

The first trial for this control method is done with walking on flat grounds. This scenario was very helpful for tuning some parameters of our simulation, including those presented in subsection III-B. In the simulation the robot walked on both short and long distances, provided we use the following gains for our various tasks, (empty cells stand for "all axes"):

Task	Type	Axis	Weight
CoM	Position, Velocity	x,y	200
		z	20
	Angular Momentum	x,y	1
Posture			2
Target	Position		20
	Orientation		150

We use a lower weight for the CoM tracking task on the z-axis otherwise the geometrical constraints of the robot (e.g. joint limits) conflict with tracking the CoM along that direction. It could seem strange that we do not impose a high weight on this particular task, as it determines a good part of our model, but the error (being low) does not disturb the closed-loop control.

This first simulation revealed that we were indeed capable of computing our lower-level control in less than 5ms, and our CoM trajectory in about 20ms, when using our default setting of a 3-second window sampled into 20 points. Experiments show that in our hardware, the number of points should not exceed 40 to keep the computation fast, and that the maximum interval length should be less than 250ms to keep a good sampling of our trajectory. We also found out that the minimum window length is around 3s to preview at least through the next stance.

B. Corniche

This scenario was designed to demonstrate multi-contact capabilities of our planner/controller. The environment consists of a flat ground that is reduced to a narrow ridge, wide of about 15cm, shown on Figure 3. To be able to cross it, we added a support plank on the left side. We disabled the momentum tracking task because the preview control was able to generate a very low-momentum trajectory while the robot has to bend forward to take the contact with his left hand. Thus, the conflicting tasks resulted in failures to properly position the hand in the given ad-hoc computed time.

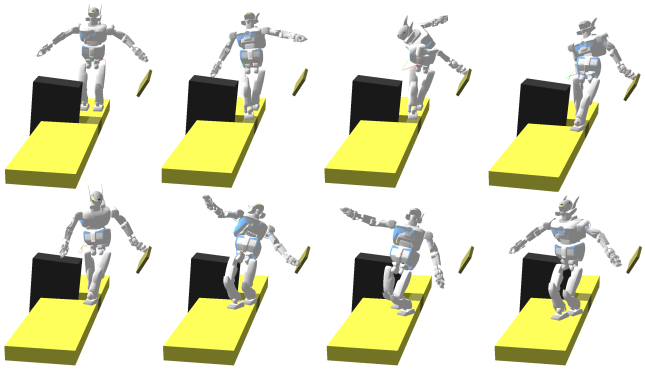

Fig. 3. HRP-2 going over a ridge: contact stances and transitions are illustrated.

C. Walking with an object on uneven terrain

As we use both hands to carry a 2kg object, shown in red on Figure 4, it is difficult to present a scenario showing at the same time the multi-contact capabilities and object manipulation. The setting is made for walking on a succession of non-coplanar surfaces while wielding the object. This trial consists of a first stair step, followed by a slope and ending on a flat floor. This trial assessed our approach to modify the mass used by the preview while walking but showed that we had to perform one modification to our general approach.

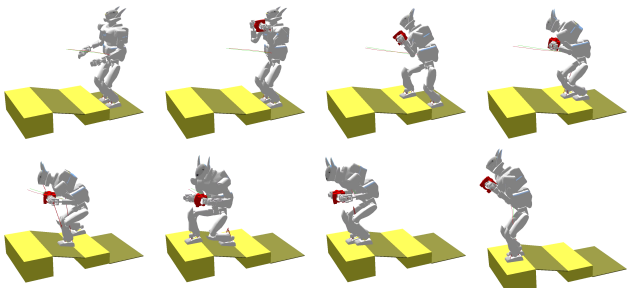

Fig. 4. HRP-2 Walking on uneven ground while carrying an object.

While holding the object, it is necessary to slightly penalize its use to regulate the CoM position. Otherwise, the robot would use this easy-to-move weight to follow the CoM, hence being late on the desired position of the robot. As this delay accumulates, the robot ends up in a position where it needs to exert tremendous torques to reach the CoM target in time, resulting in instability and toppling.

In terms of results, the robot executes a correct motion, although not fully 'human-like', mainly because the robot crouches down more than we would do, as shown by the dip at the end of the trajectory. Although HRP-2 is quite strong in the arms, lifting heavy weights is usually done using handles rather than two small unilateral contacts, as in this paper. On top of this the loaded robot's CoM is higher than unloaded but we still target the stable end CoM computed by the planner, resulting in an increased tendency to crouch.

In this experiment, we also plotted on Figure 5 the uncontrolled momentum derivative, $\dot{L}_z$ computed by two methods

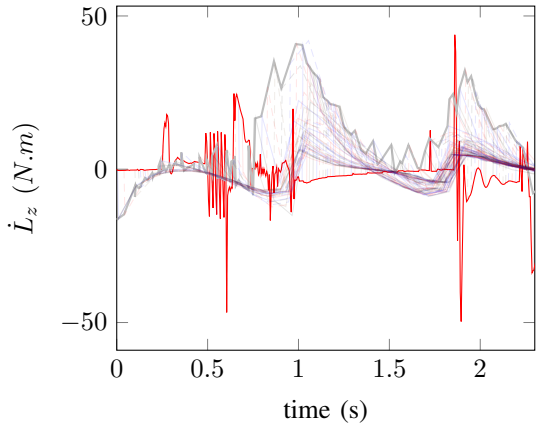

Fig. 5. Uncontrolled momentum derivative $\dot{L}_z$ computed on the whole-body model (thin red) and reduced flywheel model (thick grey) with superposition of computed trajectories (multicolor transparent)

over the two first seconds. The first one is the momentum of the full robot, computed by the controller. We plot the finite difference derivative of this quantity. The second one is computed by the preview control on the flywheel model. Using the fact that $\dot{L}_z$ depends (non-linearly) on the state X and control vector U, we can compute it *a posteriori*. This figure shows that the momentum derivative computed by the preview is almost always lower than the one of the real robot, and does not present the same variations.

VI. Conclusion and future work

We presented a multi-purpose control scheme to realize a variety of multi-contact dynamic motions eventually holding (with both arms) another rigid body (motion with sustained external forces). We also presented practical heuristics to partially achieve automatic parametrization of the model preview control in multi-contact motion. The simulated scenarios revealed that although attractive, many limitations would jeopardize a robust implementation on a real humanoid robot. First of all, despite a heuristic tuning of the timing, it is far from being a solved issue. For now, we associate a feasible timing to each contact, but we do not have a criteria to recompute them in case of imminent failure due to conflicting tasks. Second, the moment around z may produce a bad behavior. However, since this momentum can be computed from the reduced model, it can be send as an objective to be tracked at best by the low-level task QP controller.

To sum-up, this study reveals that seeking for linear models based on reduce models come with drawbacks that are difficult to circumvent for general purpose multi-contact motions. Time parameterization of the contact formation/release and their transition phases can hardly be left for tuning.

Future work consists in either incrementally go toward nonlinear formulation but still keeping a two phase reduced model or work toward whole-body nonlinear MPC using best of GPGPU parallelization and solver customization. We clearly do not have a clear answer on what best to go with.

VII. Acknowledgements

This work is partially supported by the FP7 EU RoboHow.Cog, FP7 KoroiBot and the JSPS Grant-in-Aid B No 25280096.

References

[1] K. Nagasaka, T. Fukushima, and H. Shimomura, "Whole-body control of a humanoid robot based on generalized inverse dynamics and multi-contact stabilizer that can take acount of contact constraints," in *Robotics Symposium (In Japanese)*, vol. 17, March 2012.

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

[3] L. Saab, O. E. Ramos, F. ois Keith, N. Mansard, P. Souères, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, April 2012.

[4] C. Collette, A. Micaelli, C. Andriot, and P. Lemerle, "Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts," in *IEEE/RAS International Conference on Humanoid Robots*, Pittsburgh, PA, November 29 - December 1 2007, pp. 81–88.

[5] J. Salini, S. Barthélemy, and P. Bidaud, *LQP-based controller design for humanoid Whole-body motion*. Springer, 2010, pp. 177–184.

[6] C. Ott, M. A. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," in *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011, pp. 26–33.

[7] K. Bouyarmane and A. Kheddar, "Multi-contact stances planning for multiple agents," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, 9-13 May 2011, pp. 5246–5253.

[8] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[9] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, 06 2014.

[10] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, August-September 2013.

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

[12] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust Physics-Based Locomotion Using Low-Dimensional Planning," *ACM Transactions on Graphics*, vol. 29, no. 3, 2010.

[13] K. Bouyarmane and A. Kheddar, "Humanoid Robot Locomotion and Manipulation Step Planning," *Advanced Robotics*, vol. 26, no. 10, pp. 1099–1126, 2012.

[14] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[15] K. Bouyarmane, J. Vaillant, F. Keith, and A. Kheddar, "Exploring humanoid robots locomotion capabilities in virtual disaster response scenarios," in *IEEE-RAS International Conference on Humanoid Robots*, Businness Inoovation Center, Osaka, Japan, November 2012.

[16] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Fransico, CA, 25-30 September 2011.

[17] J. Pan and D. Manocha, "GPU-based parallel collision detection for fast motion planning," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 187–200, 2012.

[18] J. Schulman, J. Ho, A. Lee, I. Awwal, and P. A. Henry Bradlow, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems IX*, 2013.