2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Humanoid Whole-Body Planning for Loco-Manipulation Tasks

Paolo Ferrari, Marco Cognetti, Giuseppe Oriolo

Abstract— **We consider the problem of planning whole-body motions for humanoids that must execute loco-manipulation tasks, i.e., manipulation tasks that implicitly require a locomotion phase. The proposed planner builds a tree in configuration-time space by concatenating feasible, collision-free whole-body motions that realize a succession of CoM movement primitives and, at the same time, the assigned manipulation task. To obtain fluid, natural motions we identify three zones of operation, i.e, locomotion, loco-manipulation and manipulation, and we carefully design a mechanism that allows to synchronize the two tasks. The proposed method has been implemented in V-REP for the NAO humanoid and successfully tested in various scenarios of increasing complexity.**

I. Introduction

Humanoid robots have been the subject of constantly increasing attention in recent years. The first focus for this kind of robots was how to achieve a stable and efficient locomotion. Nowadays, the main challenge is to generate whole-body motions aimed at fulfilling complex task.

Planning the motion for humanoids is challenging for a number of reasons. First, these robots have many degrees of freedom and therefore a high-dimensional planning space. Second, they are not free-flying systems in their configuration space: feasible motions should be appropriately generated, also taking into account several kinematic and/or dynamic constraints. Furthermore, equilibrium (either static or dynamic) should be maintained at all times. Due to the difficulties of the problem, it is usually simplified by taking some simplifying assumptions on either the environment or the robot geometry (see, e.g., [1], [2], [3], [4], [5], [6], [7]).

Assigning a task (e.g., grasp an object) that the robot has to execute further increases the difficulty of the planning problem. The most common approach for dealing with tasks is kinematic control, e.g., see [8], [9]. This framework was used in [10] to generate footsteps for humanoid robots. Despite its efficiency, kinematic control is a local technique well suited for designing reactive behaviors but usually outperformed by full-fledged planning techniques.

Assuming that the scene geometry is known, three main approaches for task-oriented planning can be found in the humanoid literature: (i) separating locomotion from manipulation [11], [12]; (ii) planning statically stable collision-free trajectories, that are converted to dynamically stable motions in a second stage [13]; (iii) achieving acyclic locomotion and manipulation through whole-body contact planning [14].

In [15], we have introduced a completely different approach that solves task-constrained planning problems without separating locomotion (footsteps) from task execution. Our framework automatically generates walking motions, if needed to complete the task. We extended this work in [16] and [17], where we presented a planner hinged on the concept of CoM movement primitives, defined as precomputed trajectories of the CoM that are associated to specific actions. This planner is able to deal with tasks expressed as either a trajectory or a set-point. In the latter case, the task was activated only when the robot enters a sphere placed around the set-point. However, in the case of reaching tasks this rudimentary activation strategy may result in unnatural motions.

In this paper, we are particularly interested in manipulation problems where the humanoid must bring a specified hand to a certain position. This position will in general be outside the workspace that the humanoid can access with a simple reaching motion from its initial configuration: to complete the task, the robot will necessarily have to take some steps. Since this kind of manipulation task implicitly requires locomotion, we refer to them as *loco-manipulation tasks*.

The generation of reaching motions has been addressed through machine learning techniques. In [18], a bidirectional RRT uses stored plans to guide the growth of the trees; however, the desired set-point is within the robot workspace so that no locomotion phase is necessary. The approach in [19] relies on humanoid imitation of visually acquired human motions; clearly, the acquisition phase must be repeated in new planning scenarios.

Here, we present a method for generating humanoid motions which is specifically aimed at loco-manipulation tasks. To this end, we introduce a randomized planner that builds a tree in configuration-time space by concatenating feasible, collision-free whole-body motions realizing a succession of CoM movement primitives and, at the same time, the assigned manipulation task. To obtain fluid, natural motions we identify three zones of operation, i.e, locomotion, loco-manipulation and manipulation, and we carefully design a mechanism that allows to synchronize the two tasks. To accomplish more complex tasks, we also extend the catalogue of CoM movements used in [16], [17] by adding more primitives. Finally, since enlarging the catalogue may increase the planning time, we propose a simple weighting mechanism aimed at speeding up the solution.

The paper is organized as follows. In the next section, we shall introduce a humanoid motion model and provide a formulation of our planning problem. Operation zones and the mechanism for synchronizing locomotion and manipulation tasks are introduced in Sect. III. The proposed motion planner algorithm is described in detail in Sect. IV. V-REP planning experiments for the NAO humanoid robot

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy. E-mail: {cognetti, oriolo}@diag.uniroma1.it. This work is supported by the EU H2020 COMANOID project.

are illustrated in Sect. V. Conclusions and future research directions are briefly discussed in Sect. VI.

II. Problem Formulation

Before formulating our motion planning problem, we recall the humanoid motion model introduced in [16], [17].

We define the configuration of the humanoid as

$$\boldsymbol{q} = \begin{pmatrix} \boldsymbol{q}_{\mathrm{CoM}} \\ \boldsymbol{q}_{\mathrm{jnt}} \end{pmatrix},$$

where $\boldsymbol{q}_{\mathrm{CoM}} \in SE(3)$ is the world pose (position and orientation) of a reference frame attached to the Center of Mass (CoM) and $\boldsymbol{q}_{\mathrm{jnt}} \in \mathcal{C}_{\mathrm{jnt}}$ is the n-vector of joint angles.

With our planner, the CoM movements will be generated by patching subtrajectories extracted by a precomputed catalogue of *CoM movement primitives*. These primitives represent elementary humanoid motions, such as stepping, crouching, and so on. Each primitive has a given duration and may specify trajectories for other points of the robot in addition to the CoM: for example, a stepping primitive will include a swing foot trajectory. Once a primitive has been selected, joint motions will be chosen so as to execute it while satisfying other requirements.

This planning approach is reflected in the hybrid (partly algebraic, partly differential) motion model

$$\boldsymbol{q}_{\mathrm{CoM}}(t) = \boldsymbol{q}_{\mathrm{CoM}}(t_k) + \boldsymbol{A}(\boldsymbol{q}_{\mathrm{CoM}}(t_k))\, \boldsymbol{u}_{\mathrm{CoM}}(t) \quad (1)$$

$$\dot{\boldsymbol{q}}_{\mathrm{jnt}}(t) = \boldsymbol{v}_{\mathrm{jnt}}(t), \quad (2)$$

where $t \in [t_k, t_k + T_k]$, with T_k the duration of the current CoM primitive. In this model, $\boldsymbol{A}(\boldsymbol{q}_{\mathrm{CoM}}(t_k))$ is the transformation matrix between the CoM frame at t_k and the world frame, $\boldsymbol{u}_{\mathrm{CoM}}(t)$ is the CoM pose displacement at t relative to t_k (as specified by the primitive), and $\boldsymbol{v}_{\mathrm{jnt}}(t)$ is the vector of joint velocity commands (which must obviously be compatible with the primitive itself).

We consider a basic manipulation problem where the humanoid must bring a specified hand (henceforth referred to as the *end-effector*) to a certain position[1], e.g., for grasping an object. This position will in general be outside the workspace that the humanoid can access with a simple reaching motion from its initial configuration: to complete the task, it will then be necessary to perform also stepping motions. These motions will be automatically generated by our motion planner, which will therefore convert the original manipulation task into a *loco-manipulation* task.

Denote by $\boldsymbol{y}_M = \boldsymbol{f}(\boldsymbol{q})$ the position variables of the chosen end-effector, and by $\boldsymbol{y}_M^*$ the desired set-point. A solution to our motion planning problem consists of a whole-body motion $\boldsymbol{q}_{\mathrm{jnt}}(t)$, $t \in [t_{\mathrm{ini}}, t_{\mathrm{fin}}]$ that satisfies four requirements:

- R1 The assigned set-point is reached at a finite time t_{fin}.
- R2 Collisions with workspace obstacles and self-collisions are avoided.
- R3 Position and velocity limits on the joints are respected.
- R4 The robot is in equilibrium at all times.

Note that in our formulation t_{fin} is not assigned and will be determined by the planner.

[1]We do not include orientation in the task for simplicity, but the proposed planner can be directly used also in that case.

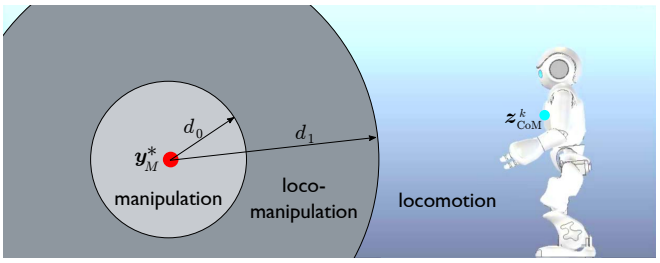

Fig. 1. Operation zones defined by our planner: *locomotion* (no manipulation set-points are used), *loco-manipulation* (local manipulation set-points are used to synchronize the end-effector with the CoM) and *manipulation* (the global manipulation set-point $\boldsymbol{y}_M^*$ is used).

III. Synchronizing Locomotion and Manipulation

The proposed planner, to be described in Sect. IV, has at its core a *motion generator* that is based on kinematic control. Within this well-known framework, set-point tasks are usually taken into account in two alternative ways:

1) The task is always active. In our case, this approach would generate unnatural motions of the humanoid, especially when the initial error is large. In fact, the robot will try to extend its arm from the very beginning. This is also likely to push some joint close to their limit, thus making motion generation more difficult.
2) The task is only activated when the robot is sufficiently close to the desired set-point. This strategy eliminates the above behavior but has two drawbacks. First, motion generation is not always driven by the task, resulting in inefficient exploration of the configuration space. Second, the overall movement will not be fluid due to the abrupt task activation.

We propose a conceptually intermediate approach. Let d be the distance between the robot CoM and the set-point $\boldsymbol{y}_M^*$, and define three zones of operation (see Fig. 1):

- *Locomotion* zone ($d_1 < d$). The robot is far from $\boldsymbol{y}_M^*$ and its end-effector is not driven by the manipulation task.
- *Loco-manipulation* zone ($d_0 < d < d_1$). At an intermediate distance, *local set-points* are used to synchronize manipulation with locomotion.
- *Manipulation* zone ($d < d_0$): At a close distance, the robot performs a reaching motion while remaining in double support.

While the threshold d_1 only determines the extension of the loco-manipulation zone, d_0 must be chosen with care so as to guarantee that inside the manipulation zone the robot can reach $\boldsymbol{y}_M^*$ easily.

The proposed planner generates motions by iteratively expanding a tree in configuration-time space. At each iteration, the planner invokes the motion generation module, which works in two stages. First, a CoM movement primitive is chosen from the catalogue; this defines the current *locomotion task*. In the second stage, kinematic control is used to compute a feasible joint motion that realize a specific combination of the current locomotion task and manipulation

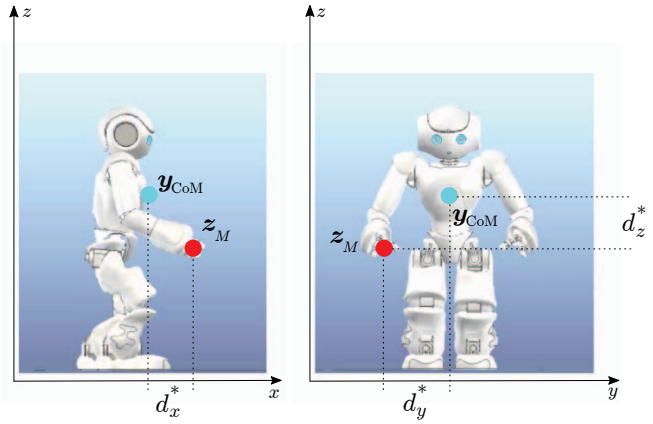

Fig. 2. The definition of the target displacement vector $\boldsymbol{d}^*$ between the end-effector (here, the right hand) and the CoM of the robot is made with reference to a comfortable reference posture.

task depending on the zone of operation.

In the locomotion zone, only the locomotion task is active, while the manipulation task is not explicitly considered. In the manipulation zone, both the current locomotion task and the global manipulation task $\boldsymbol{y}_M^*$ are active. In the loco-manipulation zone, the idea is to define local-set points for the manipulation task so that is synchronized with the current locomotion task. Let us examine in detail how this is done.

Consider a generic iteration where the tree vertex selected for expansion is associated to time t_k. Let $\boldsymbol{u}_{\mathrm{CoM}}$ be the CoM displacement associated to the chosen primitive, and compute the resulting CoM position $\boldsymbol{z}_{\mathrm{CoM}}$ over $[t_k, t_{k+1}]$ by using eq. (1) and extracting the Cartesian part. Denote by $\boldsymbol{z}_{\mathrm{CoM}}^k$, $\boldsymbol{z}_{\mathrm{CoM}}^{k+1}$ the CoM position respectively at t_k, t_{k+1}. In the loco-manipulation zone, a local set-point for the manipulation task (i.e., for the robot end-effector) is computed as:

$$\boldsymbol{y}_M^{k+1} = \boldsymbol{z}_{\mathrm{CoM}}^{k+1} + \boldsymbol{d}^k + \boldsymbol{K}_d(\boldsymbol{d}^* - \boldsymbol{d}^k), \tag{3}$$

where $\boldsymbol{d}^k = \boldsymbol{y}_M^k - \boldsymbol{z}_{\mathrm{CoM}}^k$ and $\boldsymbol{K}_d > 0$ is a diagonal gain matrix. Use of eq. (3) will force the humanoid to move its end-effector and the CoM in a coordinated fashion over the iterations, progressively bringing the displacement between them to a target value $\boldsymbol{d}^*$. The choice of $\boldsymbol{d}^*$ (see Fig. 2) can be made by placing the humanoid robot in a comfortable reference posture that is considered appropriate for entering the manipulation zone and performing the final part of its assigned task.

At the entrance of the manipulation zone, the set-point for the manipulation task is set back to the global set-point $\boldsymbol{y}_M^*$. The jump in the set-point might cause a jerking motion due to the sudden increase of joint velocities. To smooth the transition, we replace (3) with

$$\boldsymbol{y}_M^{k+1} = \boldsymbol{z}_{\mathrm{CoM}}^{k+1} + \boldsymbol{d}^k + \boldsymbol{K}_d(\boldsymbol{d}_k^* - \boldsymbol{d}^k), \tag{4}$$

where $\boldsymbol{d}_k^* = \alpha\,\boldsymbol{d}^* + (1-\alpha)(\boldsymbol{y}_M^* - \boldsymbol{z}_{\mathrm{CoM}}^k)$ and α is a sigmoid function

$$\alpha = \frac{1}{1 + \exp(-12\frac{\|\boldsymbol{z}_{\mathrm{CoM}}^k - \boldsymbol{y}_M^*\|^2 - d_0}{d_1 - d_0} + 6)}.$$

With this update law, the difference vector $\boldsymbol{y}_M^{k+1} - \boldsymbol{z}_{\mathrm{CoM}}^k$ (the displacement between the robot CoM and the local-set point) smoothly varies from $\boldsymbol{d}^*$ (the target displacement) to d_0 (the displacement between the robot CoM and the global set-point at the entrance of the manipulation zone) as the robot approaches the manipulation zone. This results in a natural reaching motion that begins in the loco-manipulation zone and fluently continues in the manipulation zone.

No smoothing is used at the boundary between the locomotion and the loco-manipulation zone, because the robot end-effector is not performing any task in the former, and therefore no discontinuity is experienced in the transition to the latter. Finally, note that the above mechanism for synchronizing locomotion with manipulation can also be used in a pure kinematic control framework.

IV. The Planner

As already mentioned, our motion planner works in an iterative fashion, building a tree in configuration-time space. To this end, it uses a catalogue U of N *CoM movement primitives*. All these primitives satisfy the R4 requirement (humanoid equilibrium) by construction. The choice of primitives may change depend on the considered planning scenario; in any case, a richer set will result in a more versatile planner. A typical U will include various stepping motions and possibly more complex movements, such as crouching, crawling, and so on.

An indispensable element of any catalogue U is `free_CoM`, a primitive that leaves the CoM free to move with the constraint that both feet should remain fixed and the robot should maintain equilibrium. This primitive, which is essential in the manipulation zone, has an arbitrary duration that will typically be determined by the time needed to reach the local set-point. However, static equilibrium is not guaranteed in `free_CoM` and must be explicitly checked.

A vertex $v = (\boldsymbol{q}, \boldsymbol{w})$ consists of a configuration $\boldsymbol{q}$, with an associated time instant (not explicitly displayed), and a set of weights $\boldsymbol{w} = \{w^1, \ldots, w^N\}$. An edge represents a whole-body motion that connects two adjacent vertexes and is *feasible*, i.e., satisfies requirements R2-R4.

The tree $\mathcal{T}$ is rooted at $v_{\mathrm{ini}} = (\boldsymbol{q}(t_{\mathrm{ini}}), \boldsymbol{w}_{\mathrm{ini}})$, with $\boldsymbol{w}_{\mathrm{ini}} = (1/N, \ldots, 1/N)$. The algorithm uses of a *task compatibility* metric $\gamma(\boldsymbol{q}, \boldsymbol{y}_M^*)$ that measures the compatibility of a configuration $\boldsymbol{q}$ with respect to the task set-point $\boldsymbol{y}_M^*$. For example, γ may be defined as the inverse of the distance between $\boldsymbol{y}_M^*$ and the robot CoM at $\boldsymbol{q}$.

A generic iteration of the planner (see Algorithm 1) starts by selecting a random vertex $v_{\mathrm{near}} = (\boldsymbol{q}_{\mathrm{near}}, \boldsymbol{w}_{\mathrm{near}})$ of $\mathcal{T}$ with probability[2] $\gamma(\cdot, \boldsymbol{y}_M^*)$; call t_k the associated time instant. The motion generator is then invoked.

The first step of motion generation (see Procedure 1) consists in extracting a random primitive from U using the weights in $\boldsymbol{w}_{\mathrm{near}}$ as probabilities. Let $\boldsymbol{u}_{\mathrm{CoM}}$ be the CoM displacement associated to this primitive, and T_k its duration. Using eq. (1), the reference pose of the CoM frame $\boldsymbol{q}_{\mathrm{CoM}}(t)$

[2]Biasing the sampling process with γ guarantees that the expansion of the tree will be directed towards $\boldsymbol{y}_M^*$ even during the locomotion phase, when the manipulation task is not explicitly active.

Algorithm 1: Planner

root the tree $\mathcal{T}$ at $v_{\rm ini} = (\boldsymbol{q}(t_{\rm ini}), (\frac{1}{N}, \ldots, \frac{1}{N}))$;
$j \leftarrow 0$;
repeat
  $j \leftarrow j + 1$;
  select a random vertex $v_{\rm near}$ from $\mathcal{T}$ with probability $\gamma(\cdot, \boldsymbol{y}^*_M)$ and retrieve the associated time instant t_k;
  $[\boldsymbol{q}_{\rm new}, \overline{\boldsymbol{q}_{\rm near}\boldsymbol{q}_{\rm new}}, t_{k+1}] \leftarrow$ MotionGeneration($v_{\rm near}, t_k$);
  if $\boldsymbol{q}_{\rm new} \neq \emptyset$ **then**
    // *vertex generation successful*;
    $\boldsymbol{w}_{\rm new} \leftarrow \left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$;
    $v_{\rm new} \leftarrow (\boldsymbol{q}_{\rm new}, \boldsymbol{w}_{\rm new})$;
    add vertex $v_{\rm new}$ and edge $\overline{\boldsymbol{q}_{\rm near}\boldsymbol{q}_{\rm new}}$ to $\mathcal{T}$;
  else
    // *vertex generation unsuccessful*;
    update the weights in $v_{\rm near}$ according to (8–9);
  end
until $\boldsymbol{f}(\boldsymbol{q}_{\rm new}) = \boldsymbol{y}^*_M$ **or** $j =$ MAX_IT;

is computed, for $t \in [t_k, t_{k+1}]$, with $t_{k+1} = t_k + T_k$. This, together with the associated reference motion of the feet, defines the current locomotion task. At this point, depending on which zone is associated to $v_{\rm near}$ (this information may be stored with the vertex at the time of its creation), the local set-point for the manipulation task is computed as explained in the previous section. This will be null in the locomotion zone, $\boldsymbol{y}^*_M$ in the manipulation zone, and given by (4) in the loco-manipulation zone.

The second part of motion generation is the computation of a whole-body motion that starts from $\boldsymbol{q}_{\rm near}$, executes the current locomotion task, reaches the local manipulation set-point (if it is not null), and complies with requirements R2-R4. To this end, we have used a task-priority approach [20]:

$$\boldsymbol{v}_{\rm jnt} = \boldsymbol{J}_L^\dagger \dot{\boldsymbol{y}}'_L + \boldsymbol{P}_L (\boldsymbol{J}_M \boldsymbol{P}_L)^\dagger (\dot{\boldsymbol{y}}'_M - \boldsymbol{J}_M \boldsymbol{J}_L^\dagger \dot{\boldsymbol{y}}'_L) + \boldsymbol{P}_{LM}\, \boldsymbol{v}_0, \tag{5}$$

Here, $\boldsymbol{y}_L$ is the (primary) locomotion task, $\boldsymbol{J}_L$ its Jacobian and $\boldsymbol{P}_L = \boldsymbol{I} - \boldsymbol{J}_L^\dagger \boldsymbol{J}_L$; $\boldsymbol{y}_M$ is the (secondary) manipulation task, $\boldsymbol{J}_M$ its Jacobian and $\boldsymbol{P}_{LM} = \boldsymbol{P}_L - (\boldsymbol{J}_M \boldsymbol{P}_L)^\dagger (\boldsymbol{J}_M \boldsymbol{P}_L)$. Also, we set $\dot{\boldsymbol{y}}'_L = \dot{\boldsymbol{y}}^*_L + \boldsymbol{K}_L \boldsymbol{e}_L$ and $\dot{\boldsymbol{y}}'_M = \boldsymbol{K}_M \mathrm{sign}(\boldsymbol{e}_M)$, where $\boldsymbol{e}_L = \boldsymbol{y}^*_L - \boldsymbol{y}_L$, with $\boldsymbol{y}^*_L$ the reference value of $\boldsymbol{y}_L$, and $\boldsymbol{e}_M = \boldsymbol{y}^{k+1}_M - \boldsymbol{y}_M$. Finally, $\boldsymbol{K}_L$ and $\boldsymbol{K}_M$ are positive definite gain matrices; in particular, we let

$$\boldsymbol{K}_M = \mathrm{diag}\{\|\boldsymbol{y}^{k+1}_M - \boldsymbol{y}^k_M\| / T_k\} \tag{6}$$

where $\boldsymbol{y}^k_M = \boldsymbol{f}(\boldsymbol{q}_{\rm near})$ is the end-effector position at t_k. The definition of $\dot{\boldsymbol{y}}'_M$ and $\boldsymbol{K}_M$ guarantees that the local set-point $\boldsymbol{y}^{k+1}_M$ is achieved exactly at t_{k+1}.

The choice of the null-space vector $\boldsymbol{v}_0$ in (5) is arbitrary. For further exploration of the solution space, we choose

$$\boldsymbol{v}_0 = -\eta \nabla_{\boldsymbol{q}_{\rm jnt}} H(\boldsymbol{q}_{\rm jnt}) + \boldsymbol{v}_{\rm rnd}, \tag{7}$$

where η is a positive stepsize, $H(\boldsymbol{q}_{\rm jnt})$ is a cost function and $\boldsymbol{v}_{\rm rnd}$ is a bounded-norm vector which is randomly generated using uniform probability. For example, $H(\boldsymbol{q}_{\rm jnt})$ may be chosen so as to maximize the available joint range.

The joint trajectories generated by the kinematic control law (5) are continuously checked for collisions (requirement R2) and for violation of position/velocity joint limits (requirement R3). If `free_CoM` is the current CoM primitive, static equilibrium of the humanoid is explicitly checked. If no violation occurs, the integration reaches t_{k+1}, generating $\boldsymbol{q}_{\rm new}$. Control goes back to the main planner that adds to $\mathcal{T}$ a new vertex $v_{\rm new} = (\boldsymbol{q}_{\rm new}, \boldsymbol{w}_{\rm new})$, with $\boldsymbol{w}_{\rm new} = (1/N, \ldots, 1/N)$, together with the edge (i.e., the whole-body motion) joining $v_{\rm near}$ with $v_{\rm new}$.

Procedure 1: MotionGeneration($v_{\rm near}$, t_k)

extract $\boldsymbol{q}_{\rm near}$ and $\boldsymbol{w}_{\rm near}$ from $v_{\rm near}$;
select from U a CoM primitive $\boldsymbol{u}^k_{\rm CoM}$ with probability $\boldsymbol{w}_{\rm near}$, and retrieve the associated duration T_k;
compute the current locomotion task;
define the local manipulation set-point $\boldsymbol{y}^{k+1}_M$;
repeat
  generate motion by integrating joint velocities (5);
  if collision **or** joint position/velocity limit violation **then**
    return $[\emptyset, \emptyset, \emptyset]$
  end
until $t = t_k + T_k$;
return $[\boldsymbol{q}_{\rm new}, \overline{\boldsymbol{q}_{\rm near}\boldsymbol{q}_{\rm new}}, t_k + T_k]$

If a violation occurs, motion generation is interrupted and no vertex is added to $\mathcal{T}$. In this case, the main planner performs a last operation before starting a new iteration. Assuming that the h-th CoM primitive had been selected for node expansion in the failed attempt, the weights associated are updated to $v_{\rm near}$ as follows:

$$w^h_{\rm near} = w^h_{\rm near} - \bar{w} \tag{8}$$

$$w^i_{\rm near} = w^i_{\rm near} + \frac{\bar{w}}{N-1},\ i = 1, \ldots, N,\ i \neq h, \tag{9}$$

where $\bar{w} = w^h_{\rm near}/2$. This penalizes the h-th CoM primitive and accordingly compensates the others.

V. Planning Experiments

The proposed planner has been implemented in V-REP on an Intel Core i3 running at 1.80 GHz. The chosen robotic platform is the NAO small humanoid, with the right hand as end-effector. The set of CoM movement primitives is defined as

$$U = \{U^S_{\rm CoM} \cup U^D_{\rm CoM} \cup U^C_{\rm CoM} \cup \texttt{free\_CoM}\}.$$

Here $U^S_{\rm CoM}$ is a subset of *static* steps (includes forward, backward, left and right steps) while $U^D_{\rm CoM}$ is a subsets of *dynamic* steps (includes one starting, cruise and stopping steps for various directions of motion). $U^C_{\rm CoM}$ includes instead movements for standing up and crouching (with both feet on the ground), together with stepping motions extracted from a crouched gait. Depending on the operation zone, not all the primitives in U may be selectable; in the locomotion zone, `free_CoM` can safely be ignored, whereas in the manipulation zone only `free_CoM`, standing up and crouching can be selected for motion generation.

The thresholds for the three operation zones are defined as $d_0 = 0.15$ m and $d_1 = 0.7$ m. In the loco-manipulation zone, the target displacement $\boldsymbol{d}^*$ is set to $(0.12, -0.16, -0.002)$ m.

We consider three planning scenarios. In the first (Fig. 3) the humanoid must grasp a ball placed on a low table. At the

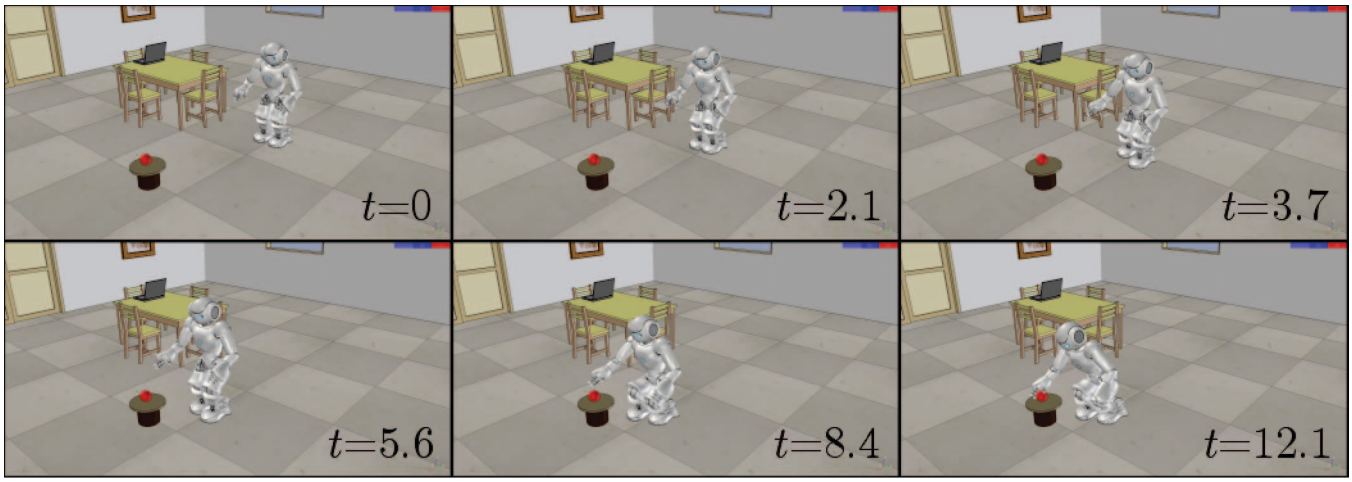

Fig. 3. Planning scenario 1: snapshots from a solution.

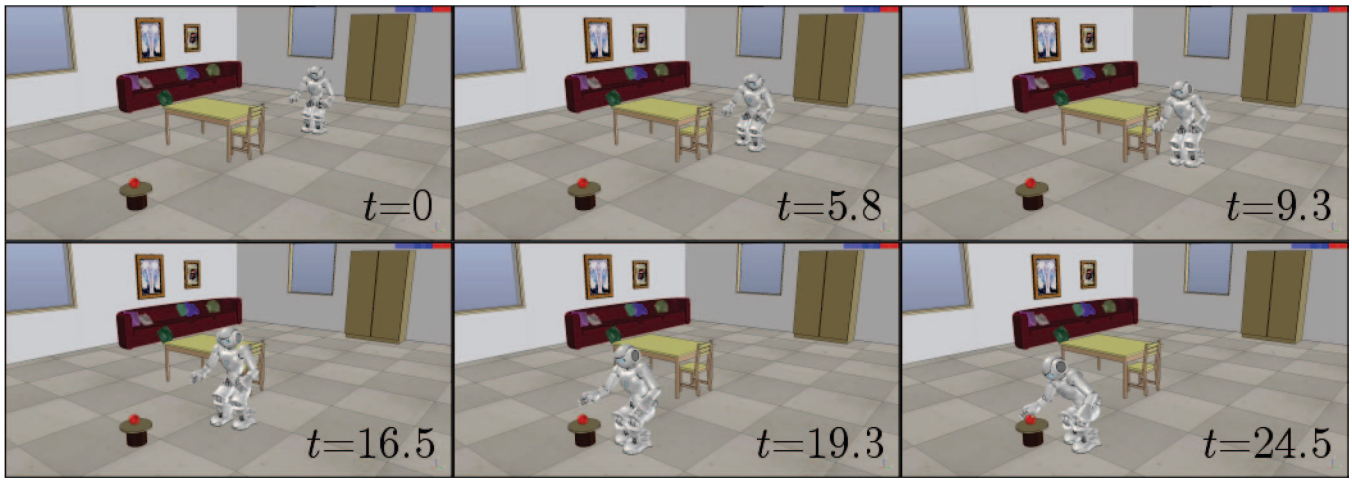

Fig. 4. Planning scenario 2: snapshots from a solution.

start (first two snapshots), the robot is in the locomotion zone and the assigned manipulation task is not directly taken into account (see footnote 2). The planner chooses a sequence of dynamic steps in the forward direction. After some steps, the robot enters the loco-manipulation zone where the movement of the right hand is synchronized with the CoM (third and fourth snapshot). Once in the manipulation zone, the robot executes first a crouching movement (fifth snapshot) and finally reaches $\boldsymbol{y}_M^*$ using the `free_CoM` primitive (last snapshot). Overall, the resulting motion of the humanoid is very fluid and natural.

The second scenario, illustrated in Fig 4, is similar to the first with the addition of some obstacles (a table and a chair) obstructing the path from the robot to the destination. The planned solution shows the robot taking some dynamic steps diagonally to avoid the obstacles and then fluidly completing the task as before.

In the third scenario (Fig 5) the robot must pass below a table to reach the table where the ball is placed. The solution shown leads the robot to the table by some forward dynamic steps, and then switches to a slightly crouched gait. Once the robot clears the table, erect posture is recovered and the manipulation task is successfully completed.

data	exp 1	exp 2	exp 3
planning time (s)	19.1	52.04	105.05
tree size (# vertexes)	32.8	97.9	135.3
motion duration (s)	12.1	24.98	23.25

TABLE I
PLANNER PERFORMANCE DATA.

The accompanying video shows dynamic playback clips of the above three solutions in which full physical simulation (including joint control) is enabled.

Table I collects some data related to the planner performance in the three scenarios. Since our planner is randomized, these data are averaged over 20 runs. Due to the absence of obstacles, solutions for the first scenario are easier to compute, have a shorter duration, and result in a smaller tree. The other two scenarios are clearly more challenging due to the obstacles in the workspace.

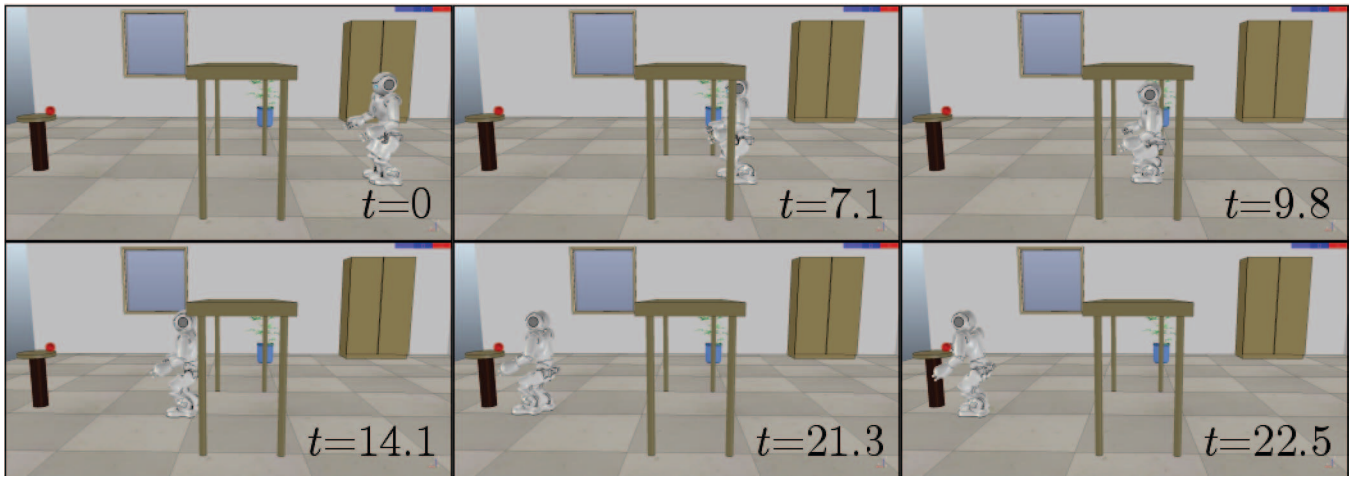

Fig. 5. Planning scenario 3: snapshots from a solution.

VI. Conclusions

We have considered the problem of planning whole-body motions for humanoids that must execute loco-manipulation tasks, i.e., manipulation tasks that implicitly require a locomotion phase. The proposed planner builds a tree in configuration-time space by concatenating feasible, collision-free whole-body motions that realize a succession of CoM movement primitives and, at the same time, the assigned manipulation task. To obtain fluid, natural motions we have identified three zones of operation, i.e, locomotion, loco-manipulation and manipulation, and we have carefully designed a mechanism that allows to synchronize the two tasks. The proposed method has been implemented in V-REP for the NAO humanoid and successfully tested in various scenarios of increasing complexity.

Future work will focus on extending this framework by adding CoM primitives in order to execute tasks requiring more complex motions, including multi-contact gaits. Another improvement would be the possibility of taking into account torque bounds, by using a second-order motion generation scheme as in [21].

References

[1] J. Kuffner, J.J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," in *2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001, pp. 500–505.

[2] 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.

[3] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *Proc. 11th Int. Symp. of Robotics Research (ISRR 2003)*, 2003.

[4] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda ASIMO humanoid," in *2005 IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 629–634.

[5] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond, "Humanoid motion planning for dynamic tasks," in *2005 5th IEEE-RAS Int. Conf. on Humanoid Robots*, 2005, pp. 1–6.

[6] J. Pettré, J.-P. Laumond, and T. Siméon, "A 2-stages locomotion planner for digital actors," in *2003 ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2003, pp. 258–264.

[7] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Trans. on Robotics*, vol. 28, no. 2, pp. 427–439, 2012.

[8] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth Int. Conf. on Advanced Robotics*, 1991, pp. 1211–1216.

[9] O. Kanoun, F. Lamiraux, and P. B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[10] O. Kanoun, J.-P. Laumond, and E. Yoshida, "Planning foot placements for a humanoid robot: A problem of inverse kinematics," *Int. J. of Robotics Research*, vol. 30, no. 4, pp. 476–485, 2011.

[11] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *Int. J. of Robotics Research*, vol. 30, no. 6, pp. 678–698, 2011.

[12] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *2013 IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 1656–1662.

[13] S. Dalibard, A. El Khoury, F. Lamiraux, A. Nakhaei, M. Taïx, and J.-P. Laumond, "Dynamic walking and whole-body motion planning for humanoid robots: An integrated approach," *Int. J. of Robotics Research*, vol. 32, no. 9–10, pp. 1089–1103, 2013.

[14] K. Bouyarmane and A. Kheddar, "Humanoid robot locomotion and manipulation step planning," *Advanced Robotics*, vol. 26, no. 10, pp. 1099–1126, 2012.

[15] M. Cognetti, P. Mohammadi, G. Oriolo, and M. Vendittelli, "Task-oriented whole-body planning for humanoids based on hybrid motion generation," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 4071–4076.

[16] M. Cognetti, P. Mohammadi, and G. Oriolo, "Whole-body motion planning for humanoids based on com movement primitives," in *2015 15th IEEE-RAS Int. Conf. on Humanoid Robots*, 2015, pp. 1090–1095.

[17] M. Cognetti, V. Fioretti, and G. Oriolo, "Whole-body planning for humanoids along deformable tasks," in *2016 IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 1615–1620.

[18] X. Jiang and M. Kallmann, "Learning humanoid reaching tasks in dynamic environments," in *2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 1148–1153.

[19] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz, "Imitating human reaching motions using physically inspired optimization principles," in *2011 11th IEEE-RAS Int. Conf. on Humanoid Robots*, 2011, pp. 602–607.

[20] S. Chiaverini, G. Oriolo, and I. D. Walker, "Kinematically redundant manipulators," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 245–285.

[21] M. Cefalo and G. Oriolo, "Dynamically feasible task-constrained motion planning with moving obstacles," in *2014 IEEE Int. Conf. on Robotics and Automation*, 2014.