Dynamic Walking and Whole-Body Motion Planning for Humanoid Robots: an Integrated Approach

Sébastien Dalibard, Antonio El Khoury, Florent Lamiraux, Alireza Nakhaei, Michel Taïx, Jean-Paul Laumond

Dynamic Walking and Whole-Body Motion Planning for Humanoid Robots: an Integrated Approach

Sébastien Dalibard Antonio El Khoury
Florent Lamiraux Alireza Nakhaei Michel Taïx
Jean-Paul Laumond *†

Abstract

This paper presents a general method for planning collision-free whole-body walking motions for humanoid robots. First, we present a randomized algorithm for constrained motion planning, that is used to generate collision-free statically balanced paths solving manipulation tasks. Then, we show that dynamic walking makes humanoid robots small-space controllable. Such a property allows to easily transform collision-free statically balanced paths into collision-free dynamically balanced trajectories. It leads to a sound algorithm which has been applied and evaluated on several problems where whole-body planning and walk are needed, and the results have been validated on a real HRP-2 robot.

1 Introduction

During the last twenty years, impressive progress has been achieved in humanoid robot hardware and control. This leads to a rising need for software and algorithms improving the usability and autonomy of those robots. One important area of research focuses on the development of robust and general motion generation techniques for safe and autonomous operation in human environments, such as offices or homes.

Motion planning for humanoid robots is challenging for several reasons. First, the computational complexity of classic motion planning algorithms is exponential in the number of Degrees of Freedom (DoFs) of the considered system, which is high for humanoid kinematic trees. Second, a humanoid robot is an under-actuated system: the DoFs that control the position and orientation

*The authors are with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse Cedex 4, France and Univ de Toulouse, LAAS, F-31400 Toulouse Cedex 4, France. Sébastien Dalibard is now with Aldebaran Robotics, France.

†This paper summarizes and extends previous work that appeared in the 9th and 11th IEEE-RAS International Conference on Humanoid Robots, 2009 and 2011.

of the whole robot in space are not directly controlled, they derive from the articular DoFs of the robot legs. Those latter should be controlled with care to guarantee dynamically balanced motions, for manipulation or navigation.

When planning collision free motions for humanoid robots, different representations of the robot and its environment can be used. The choice of the level of details of the representation indicates the difficulty of the considered problem. The simplest option consists in considering the robot as a navigating 2D shape (Pettré, Laumond & Siméon 2003), and computing obstacle avoidance in a planar model of the world. Another possibility is to compute only collision-free footsteps (Kuffner Jr, Nishiwaki, Kagami, Inaba & Inoue 2001). In complex and difficult environments, such as the one presented in Fig. 1, it can be necessary to consider exact 3D models of a humanoid robot and its environment.

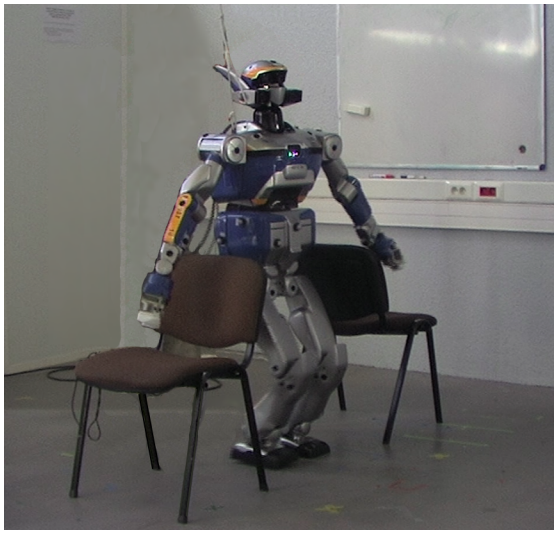

Figure 1: The robot HRP-2 passing between two chairs. In this kind of environment whole-body collision avoidance is needed during locomotion.

There are two main ways of using motion planners to generate dynamically balanced robotic motions. The more general one is to plan in a robot dynamic space, see for example (Shkolnik, Levashov, Manchester & Tedrake 2011). By taking into account both robot configuration and velocity, motions that satisfy dynamic balance constraints can be generated at a planning phase. When planning motion for humanoid robots, this is a particularly costly approach, as the size of the space to explore is augmented with the robot velocity and footprint positions. The other way is to first plan a geometric path that can be approximated by a dynamic trajectory in a second step (Yoshida, Belousov, Esteves & Laumond 2005). The approach we present in this paper falls into the second category. Some feasible dynamic motions are inherently impossible to compute with this kind of approach. For example, jumping motions cannot be generated

by a purely geometric planner.

In this paper, we present a planning algorithm that considers exact models of a humanoid robot and its environment. It is used to solve navigation and manipulation problems. Our planner is a two-step algorithm: a first collision-free path is computed in the space of quasi-statically balanced configurations, then this first path is approximated by a sequence of dynamic walking trajectories. The proof of correctness of the algorithm is based on the concept of small-space controllability. This property allows, under some assumptions, to approximate any *non necessarily admissible* path, by a sequence of admissible trajectories. In our context we prove that dynamic walking makes humanoid robots small-space controllable. Our planner is designed for perfectly modeled indoor environments, where the floor is horizontal and flat. Also, because we do not explicitly compute footprint positions at the planning stage, our planner is unable to plan motions in which the robot steps over obstacles. These limitations are discussed in the paper.

1.1 Outline

Section 2 reviews the related work and states our contribution. Section 3 presents a constrained motion planning algorithm, and its use on a humanoid robot manipulation problem. Section 4 generalizes the previous algorithm to problems that require locomotion. The generalization is well-grounded, and based on a controllability property of legged robots demonstrated in the paper. Section 5 presents some experimental results, and Section 6 discusses the limitations and potential future work of our method.

2 Related Work and Contribution

This work is based on several fields of humanoid robotics research: prioritized inverse kinematics, randomized whole-body motion planning and walk pattern generators based on the Zero-Moment Point (ZMP). This section summarizes the literature related to each of these fields.

2.1 Prioritized Inverse Kinematics

The problem of inverse kinematics for a humanoid robot, or any articulated structure, is to compute a joint position to achieve an end-effector pose. As the robots we deal with are redundant, it is natural to take advantage of this redundancy by specifying multiple tasks, potentially with different priorities. This problem has been widely studied in robotics planning and control literature, and many Jacobian-based solutions have been proposed, among which (Nakamura & Hanafusa 1986), (Siciliano & Slotine 1991), (Baerlocher & Boulic 1998) and (Khatib, Sentis, Park & Warren 2004). Obstacle avoidance can be taken into account with similar methods. To do so, one has to include the obstacles as constraints to satisfy, see for example (Kanehiro, Lamiraux, Kanoun, Yoshida

& Laumond 2008). These methods are prone to fall into local minima, thus global motion planning is needed to overcome this limitation. Note that when local methods find solutions, these are usually smoother and may look more natural. The choice of using global motion planners is justified by the need for complete algorithms. (Toussaint, Gienger & Goerick 2007) propose a motion generation method where tasks follow trajectories defined by cubic B-splines. The whole-body motion is optimized with respect to the control point positions. This method can take into account collisions with simple obstacles.

2.2 Whole-Body Motion Planning

When planning a whole-body motion for a humanoid robot, one difficult challenge is to cope with the curse of dimensionality. The complexity of motion planning is exponential in the dimension of the configuration space ($\mathcal{C}$) to explore. When dealing with high-dimensional configuration spaces, it is typically impossible to explicitly represent them, leading to the use of randomized sampling techniques to solve global planning problems. In the past fifteen years, *Probabilistic Roadmaps* (Kavraki, Svestka, Latombe & Overmars 1996) and *Rapidly exploring Random Trees* (RRT) (Kuffner & LaValle 2000) have been developed and used to solve many high-dimensional planning problems, see (LaValle 2006) and (Choset, Lynch, Hutchinson, Kantor, Burgard, Kavraki & Thrun 2005) for comprehensive overviews. When using sampling techniques on a humanoid robot, another difficulty is to take into account balance constraints, i.e. to generate random configurations on zero volume submanifolds of $\mathcal{C}$. This problem has been investigated with success during the last few years, (Berenson, Srinivasa & Kuffner 2011*b*) presents an exhaustive survey of Jacobian-based methods. Other recent contributions (Porta, Jaillet & Bohigas 2012) present sophisticated constrained motion planning techniques based on higher-dimensional continuation. Section 3 presents a simple adaptation of the RRT algorithm to constrained motion planning, that was first introduced in (Dalibard, Nakhaei, Lamiraux & Laumond 2009).

2.3 Walk Pattern Generation

Another field of humanoid robotics research is the generation of dynamically balanced walk patterns. Since the introduction of the ZMP (Vukobratovic & Juricic 1969), several methods have been proposed to generate walking motions efficiently. One way to deal with the complexity of a humanoid robot kinematic tree is to use the so-called "cart-table" simplified model (Kajita, Kanehiro, Kaneko, Fujiwara, Harada, Yokoi & Hirukawa 2003). Based on this model, planning a trajectory for the ZMP is reduced to planning a trajectory for the Center of Mass (CoM) of the robot. Given a trajectory of the CoM and footprint positions, inverse kinematics solvers can animate the whole set of DoFs of the robot to generate a dynamically balanced walk trajectory.

2.4 Collision-Free Walk Planning

Collision-free locomotion trajectories are usually obtained by simplifying the model of either the robot or the environment. By reducing a humanoid robot to a bounding volume that wraps the swaying motion, one can use a simple planar motion planner on this bounding volume and generate a valid locomotion trajectory. This strategy is used in (Pettré et al. 2003) in a computer animation context. Variants of this method include dynamic path reshaping (Yoshida et al. 2005): if collisions appear when animating the locomotion trajectory, it is locally reshaped and re-animated. This two-stage strategy does not guarantee that the locomotion trajectory can be followed or that the local reshaping will converge.

One possible simplification of the environment consists in considering obstacles at a footstep level only. (Kuffner Jr et al. 2001, Chestnutt, Lau, Cheung, Kuffner, Hodgins & Kanade 2005, Kuffner, Nishiwaki, Kagami, Inaba & Inoue 2005) use an A* algorithm to find collision-free footsteps. In (Perrin, Stasse, Baudouin, Lamiraux & Yoshida 2012), the authors compute dynamic walking motions avoiding collisions at the leg level by using an RRT algorithm.

Some planning methods for free-climbing robots (Bretl 2006) can be seen as a general way to consider quasi-static multi-step planning. They are not however directly applicable to humanoid dynamically balanced locomotion. Other recent contributions to the field of locomotion planning include algorithms considering the dynamics at the planning phase (Shkolnik et al. 2011). This leads to a growth of algorithmic complexity, particularly costly for high-dimensional systems such as humanoid robots. To the authors' knowledge such techniques have not been used on humanoid robotic platforms so far.

We show in this work that, under some assumptions, it is sufficient to plan a first path in the quasi-static configuration space of a humanoid robot, and then use the small-space controllability property to approximate this path by an admissible trajectory, i.e. a dynamically balanced walking motion. This result was first presented in (Dalibard, El Khoury, Lamiraux, Taix & Laumond 2011).

2.5 Contribution

The main contribution of this work is a whole-body motion planner for humanoid robots that computes collision-free walking trajectories, based on exact models of both robot and environment. It is used to solve manipulation tasks that may require walking. The first stage of our algorithm uses a sampling-based constrained motion planner and computes a collision-free statically balanced path for a robot which can be fixed or sliding on the ground.

Another contribution of this paper is the formal proof that dynamic walking makes humanoid robots small-space controllable, with the direct implication that this first path can always be approximated by a dynamically balanced, collision-free walking trajectory. We have implemented this well-grounded method, and the results have been validated on the HRP-2 robot.

3 Randomized Motion Planning on Constraint Manifolds

This section presents an algorithm for constrained motion planning on a sub-manifold $\mathcal{M}$ of the configuration space $\mathcal{C}$. In the particular case of a humanoid robot which is an under-actuated system, $\mathcal{C} = \mathcal{Q} \times SE(3)$, where $\mathcal{Q}$ represents the kinematic tree actuators and $SE(3)$ represents the position in space of the root of the tree. If the robot has n actuated DoFs, then $dim(\mathcal{Q}) = n$ and $dim(\mathcal{C}) = n + 6$. A configuration q of $\mathcal{C}$ is said to be valid iff, besides being collision-free, it lies on the manifold $\mathcal{M}$; we call $\mathcal{M}$ the planning manifold.

The problem solved here differs from classic approaches in two ways:

1. the set of valid configurations is defined implicitly, as the set of collision-free configurations satisfying a given set of inverse kinematics balance constraints;

2. the goal manifold $\mathcal{M}_g$ is also defined implicitly, by additional inverse kinematics constraints.

During global planning, we will consider several types of constraints for various reasons:

- Static balance: the CoM of the robot stays above the support polygon center, the two feet are horizontal on the ground.

- End-effector position and orientation: the goals of some problems presented in the experimental section of this paper are defined as a specific robot hand pose, or a gaze direction.

- Configuration task: our adaptation of randomized motion planning algorithms uses tasks defined as the distance towards a given configuration in $\mathcal{C}$. This will be detailed in the following section.

Our algorithm generalizes randomized tree expansion strategies, introduced in both (Hsu, Latombe & Motwani 1999) and (Kuffner & LaValle 2000) to constrained motion planning problems. Next subsection will recall the structure of the RRT algorithm, a popular randomized motion planning algorithm. Our method could be applied to any randomized sampling planning method, such as RRT* (Karaman & Frazzoli 2011), which is slower than RRT but generates optimal paths.

3.1 Rapidly exploring Random Trees (RRT)

The classic RRT algorithm, as presented in (Kuffner & LaValle 2000), grows a random tree inside the robot collision-free configuration space $\mathcal{C}_{free}$. Each iteration of the algorithm attempts to extend the tree by adding new vertices in the direction of a randomly selected configuration q_{rand}. Algorithm 1 shows the

Algorithm 1 RRT(q_0)

$\mathcal{T}$.Init(q_0)
for $i = 1$ to K **do**
  $q_{rand} \leftarrow$ Rand($\mathcal{C}$)
  $q_{near} \leftarrow$ Nearest($q_{rand}, \mathcal{T}$)
  Extend($\mathcal{T}, q_{near}, q_{rand}$)
end for

pseudo-code of the RRT algorithm. It takes as input an initial configuration q_0 and grows a tree $\mathcal{T}$ rooted in q_0.

One way to make the RRT algorithm more efficient is to grow trees from both the initial and goal configurations. This was first proposed in (Kuffner & LaValle 2000). Our formulation of manipulation planning does not include an explicit goal configuration, so it is not possible to directly grow a tree from the goal. To make use of the idea of growing multiple trees, we first randomly sample the goal manifold and generate several goal configurations. Then, we grow random trees from the initial configuration and the random goal configurations. The idea of generating several goals for manipulation planning was proposed in (Diankov, Ratliff, Ferguson, Srinivasa & Kuffner 2008).

Section 3.2 describes a constraint solver, Section 3.3 the goal manifold sampling, and Section 3.4 the adaptation or RRT random extensions to constrained motion planning.

3.2 Constraint Solver

We show here how a multiple constraint solver works: its purpose is to find the root q of a non-linear C^1 function $f(q)$ with a tolerance of ϵ. If we want to find a configuration on a manifold $\mathcal{M}$, $f(q)$ can be defined as a vector valued function that contains the concatenation of all constraints defining $\mathcal{M}$. Note that as the intersection of two or more manifolds is also a manifold, this constraint solver allows us also to generate configurations that lie at the intersection of several manifolds.

Algorithm 2 implements a Newton-Raphson method (Bonnans, Gilbert, Lemaréchal & Sagastizábal 2006): starting from an initial value of q, q is updated iteratively by $-\alpha \left(\frac{\partial f}{\partial q}(q)\right)^+ f(q)$, where α denotes a gain and $\left(\frac{\partial f}{\partial q}(q)\right)^+$ denotes the Moore-Penrose pseudo-inverse of the Jacobian of $f(q)$. The use of an adaptive gain α, which increases iteratively from an initial value α to a maximum value α_{max}, allows the overshoot avoidance and convergence acceleration. The update rule relies on a real factor $w \in [0, 1]$; the greater w is, the faster α will reach α_{max}. Obviously, the solver convergence depends on the initial value of q, and a bad initialization can lead to either slow convergence or failure. A cutoff number of iterations it_{max} is hence introduced to bypass these cases.

We observed that values of $\epsilon = 10^{-6}$, $\alpha = 0.1$, $\alpha_{max} = 0.95$ and $w = 0.8$ lead to good behavior, i.e. fast convergence and low failure rate. These values

are kept constant for all scenarios in this work.

Algorithm 2 `SolveConstraints`(q, f, ϵ): find q such that $f(q) = 0$

$i = 0$
while $\|f(q)\| > \epsilon$ and $i \leq it_{max}$ **do**
  // $(.)^+$ denotes the Moore-Penrose pseudo-inverse
  $q \leftarrow q - \alpha \left(\frac{\partial f}{\partial q}(q)\right)^+ f(q)$
  $i \leftarrow i + 1$
  // Make α tend toward α_{max}
  $\alpha \rightarrow \alpha_{max} - w(\alpha_{max} - \alpha)$
end while
if $\|f(q)\| \leq \epsilon$ **then**
  return q
else
  return failure
end if

3.3 Goal Manifold Sampling

The way we generate a goal configuration is the following:

1. Shoot a random configuration q_{rand} in $\mathcal{C}$ with uniform distribution.
2. Call `SolveConstraints` (Algorithm 2) on q_{rand}, with $f(q)$ defined by the intersection of the planning and goal manifolds $\mathcal{M} \cap \mathcal{M}_g$.
3. If success, check for collisions.

Fig. 2 shows resulting random configurations which satisfy both balance ($\mathcal{M}$) and reaching ($\mathcal{M}_g$) constraints for the HRP-2 robot.

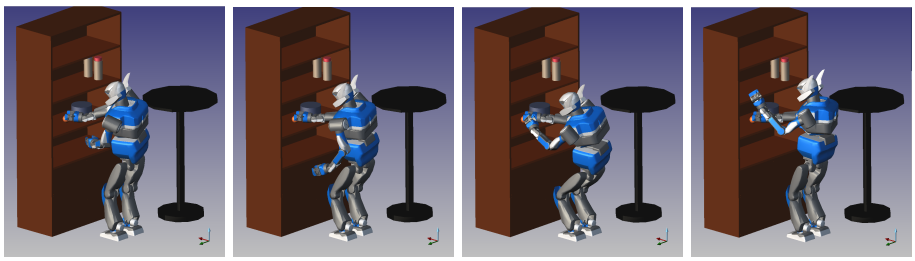

Figure 2: Random goal configurations solving a reaching task. All the configurations are balanced and collision-free, and the right hand of the character reaches the orange ball.

3.4 Random Extensions on a Constrained Manifold

Fig. 3 shows an extension of the classic RRT algorithm, from a configuration already in the tree q_{near} towards a random configuration q_{rand}.

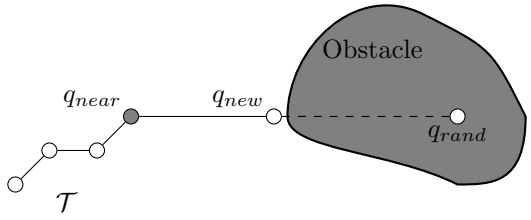

Figure 3: One step of extension of the RRT algorithm. The algorithm tries to add the longest possible edge from q_{near} towards q_{rand}, while avoiding collisions.

The equivalent random extension on a constrained manifold $\mathcal{M}$, defined by the constraint function f, starts from a valid configuration $q_{near} \in \mathcal{M}$, and extends the tree towards a random configuration q_{rand}, while keeping the constraints satisfied. Extension attempts orthogonal to $\mathcal{M}$ are useless, as newly added edges have to be included in $\mathcal{M}$. To extend in directions that follow the directions of $\mathcal{M}$, we rely on Jacobian-based inverse kinematics. Algorithm 3 presents the adaptation of the classic extend function, and Fig. 4 illustrates this extension. The idea is to first project q_{rand} on the tangent space to $\mathcal{M}$ at q_{near}. Let us call the projected configuration q'_{rand}. Let q''_{rand} be the result of a call to `SolveConstraints`(q'_{rand}, f, ϵ). It is the projection of q'_{rand} on $\mathcal{M}$. Instead of extending the tree from q_{near} towards q_{rand}, the algorithm tries to extend from q_{near} towards q''_{rand} while remaining on $\mathcal{M}$ [1]. While extending the tree, the configurations along the new edge are automatically projected onto $\mathcal{M}$. These projections are not very costly if the edge is close to the constrained manifold.

(Berenson, Srinivasa & Kuffner 2011*a*) presents a formal proof that projection-based constrained random motion planning on a fixed dimension manifold is probabilistically complete. This proof applies to our algorithm.

3.5 Example

We present in Fig. 5 an illustration of the use of randomized motion planning on complex manipulation problems. The humanoid robot HRP-2 faces shelves. It has to: (i) grasp a ball lying on a shelf, (ii) put it on a higher shelf, (iii) come back to a natural rest configuration. We can hence define three separate constrained motion planning problems where the planning manifold $\mathcal{M}$ is the static balance manifold defined in 3; the goal manifold of problem (i) is defined

[1] This presentation attempts to give a precise idea of the algorithm, without focusing on technical details. Readers interested in the actual implementation can refer to `https://github.com/laas/hpp-constrained` and `https://github.com/laas/hpp-constrained-planner`, where the corresponding open-source code is available.

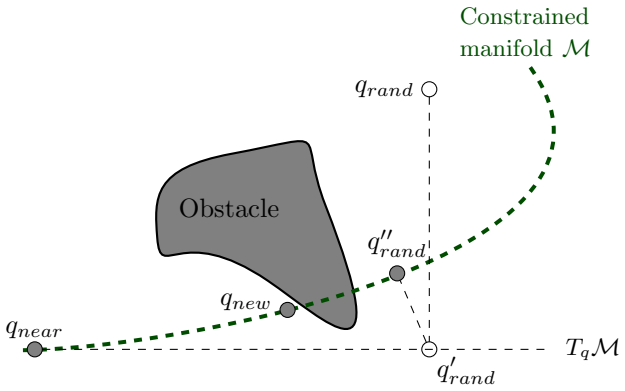

Figure 4: One step of constrained extension illustrating Algorithm 3: q_{rand} is first projected on $T_q\mathcal{M}$ the tangent space of $\mathcal{M}$. q'_{rand} is then projected onto $\mathcal{M}$. A classic RRT extension tries to go as far as possible from q_{near} towards q''_{rand} while remaining on $\mathcal{M}$. q_{new} is then returned.

Algorithm 3 `ConstrainedExtend`$(\mathcal{T}, q_{near}, q_{rand}, f, \epsilon)$

$d \leftarrow$ `Distance`(q_{near}, q_{rand})
$q \leftarrow q_{near}$
while $d > \epsilon$ **do**
  $q'_{rand} \leftarrow$ `OrthogonalProject`$(q_{rand}, T_q\mathcal{M})$
  $q''_{rand} \leftarrow$ `SolveConstraints`(q'_{rand}, f, ϵ)
  $d \leftarrow$ `Distance`(q, q''_{rand})
  $q \leftarrow q''_{rand}$
end while
$q_{new} \leftarrow$ RRT::Extend$(\mathcal{T}, q_{near}, q''_{rand})$

by a hand pose constraint (the hand must be horizontal and its position has to coincide with the ball initial position), and a gaze constraint (the robot has to look at the ball in its initial position). Similarly, the goal manifold of problem (ii) is defined by hand and gaze constraints that correspond to the position of the ball on the higher shelf. Finally, we define the rest configuration as the single goal configuration for problem (iii).

Note that for phases (i) and (iii) the ball is also considered as an obstacle. This is necessary to prevent the robot grasping hand from colliding with the ball during the approach (respectively retraction) phase.

For the two reaching motions in (i) and (ii), we first generate 8 random goal configurations (Section 3.3), then we solve the three constrained motion planning problems separately. As randomized motion planning algorithms produce, a classic shortcut method is used to optimize and shorten the paths. Extension 1 presents a video of the concatenated motion.

We have run this set of motion planning problems 20 times; results are com-

	min	max	average	average per problem
number of nodes	43.00	481.00	102.70	
goal generation time (s)	1.00	1.56	1.22	
planning time (s)	67.36	376.84	134.28	44.76

Table 1: Experimental results on 20 runs: Each run consists of 3 motion planning problems and 2 goal generations for the three phases. Time is expressed in seconds.

piled in Table 1. We have also measured the performance of `SolveConstraints` (Algorithm 2) when used to project configurations on $\mathcal{M}$; the average number of iterations is 6.5 per call, and the success rate, i.e. the ratio of the number of successfully projected configurations over the total number of calls, is above 95 percent. This success rate, high as it is, could be further improved by sampling a better initial configuration of $\mathcal{C}$, for example by introducing a heuristic bias towards statically balanced configurations. Nevertheless we choose to sample $\mathcal{C}$ uniformly-randomly for the sake of genericity.

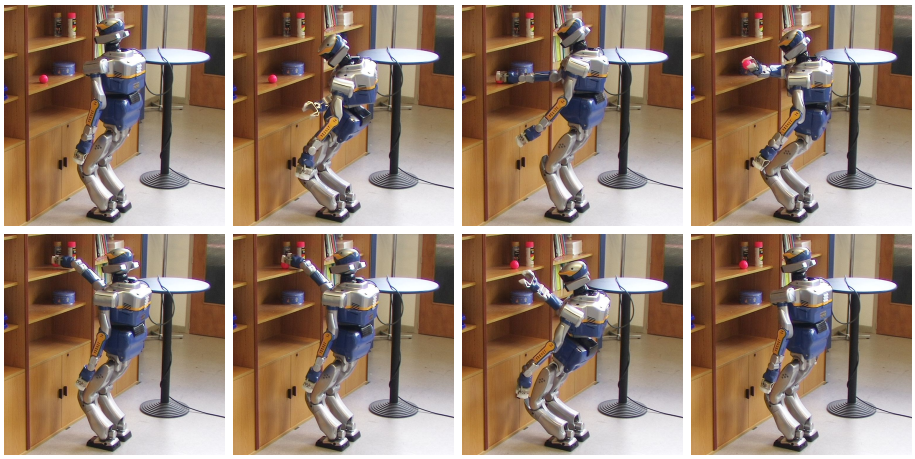

Figure 5: HRP-2 grabs a ball on a shelf, puts it on another shelf, and comes back to a rest position. Static balance constraints are enforced along the path, and the intermediary goals consisting in grasping and displacing the ball are defined implicitly as inverse kinematics constraints.

4 From Statically Balanced Paths to Dynamic Walk Trajectories

The previous section has presented a simple algorithm that solves manipulation planning problems on a given constraint manifold $\mathcal{M}$ of $\mathcal{C}$.

If we use this algorithm with static balance constraints without fixing globally the robot foot positions, it generates statically balanced paths for a robot *sliding* on the ground. Fig 6 shows an example of a whole-body collision-free path for a robot passing between two chairs. Since in reality a legged robot cannot slide on a regular floor, such paths are physically unfeasible. They are, however, easier to generate than feasible dynamic walking trajectories because only geometric constraints are considered at planning time.

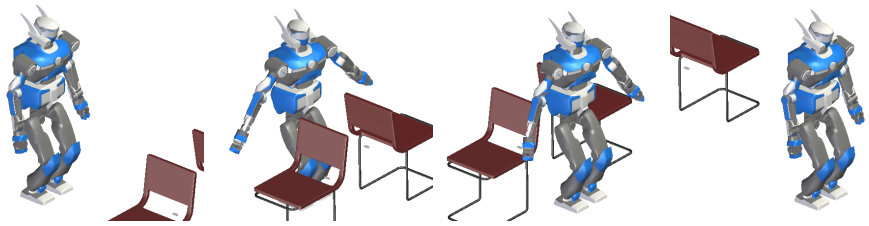

Figure 6: Collision-free statically balanced path for a humanoid robot sliding on the ground.

This section presents a *constructive* proof that any such statically balanced, collision-free path for a legged robot sliding on the ground can be approximated by a dynamically balanced, collision-free walk trajectory. The proof is based on ideas from control theory, in particular small-space controllability. It also uses the fact that balance criteria for dynamic walking are different from the ones for static balance.

Section 4.1 recalls the definition of small-space controllability and its use in motion planning. Section 4.2 proves that a dynamically walking legged robot is small-space controllable, while a quasi-statically walking legged robot is not. Section 4.3 shows how this property is used to approximate collision-free statically balanced paths by dynamic walking trajectories.

4.1 Small-Space Controllability

A robotic system is controllable if for any two configurations q_1 and q_2, there exists a trajectory going from q_1 to q_2. It is *small-space controllable* if for all configurations q, for all $\epsilon > 0$, there exists $\eta > 0$ such that all the configurations contained in the ball of center q and radius η are reachable by trajectories included in the ball of center q and radius ϵ. Fig. 7 shows an illustration of this property.

The main consequence of this property in motion planning is the following theorem, that shows how planning for dynamic systems is reduced to geometric

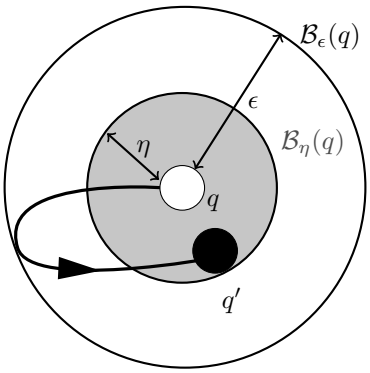

Figure 7: The small-space controllability local property: any configuration q' at a distance less than η is reachable from q by an admissible trajectory included in a ball of size ϵ.

planning thanks to the small-space controllability property:

Theorem 1. *Any collision-free path of a small-space controllable system can be approximated by a sequence of both collision-free and admissible trajectories. Thus, small-space controllability reduces trajectory planning problems to geometric path planning problems.*

Fig. 8 shows an example of collision-free path approximation by admissible collision-free sub-trajectories. The convergence of this algorithm is guaranteed by the small-space controllability property.

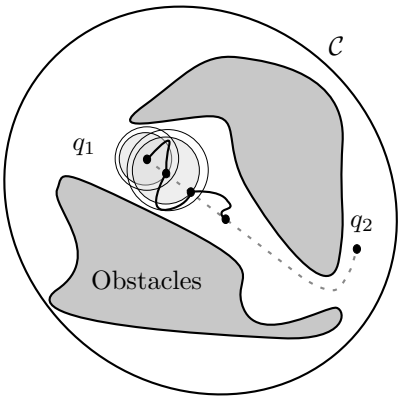

Figure 8: Small-space controllability in motion planning. A collision-free path from q_1 to q_2 is approximated by collision-free and admissible trajectories by using the local property.

This result has been long known and used in motion planning, in particular for non-holonomic systems. A detailed proof can be found in (Laumond, Jacobs, Taix & Murray 1994). We will present a sketch of the proof to give an intuition about the corresponding algorithm.

Proof of Theorem 1. Let $\mathcal{C}$ be the configuration space of a small-space controllable robot, and $\mathcal{C}_{free} \subset \mathcal{C}$ the set of collision-free configurations. We consider in-contact configurations as colliding, so $\mathcal{C}_{free}$ is an open set. Let $\tau : [0,1] \to \mathcal{C}_{free}$ be a collision-free path. Thus for all $x \in [0,1]$, $\tau(x) \in \mathcal{C}_{free}$, there exists ϵ_x such that the open ball $B(\tau(x), \epsilon_x)$ of center $\tau(x)$ and radius ϵ_x is included in $\mathcal{C}_{free}$. The small-space controllability property states that for all x, there exists $\eta_x > 0$ such that every configuration $q \in B(\tau(x), \eta_x)$ is reachable from $\tau(x)$ by a trajectory included in $B(\tau(x), \epsilon_x)$.

The set of open balls $(B(\tau(x), \eta_x))_{x\in[0,1]}$ forms an open cover of $\tau([0,1])$ which is compact. The Heine-Borel theorem (Fitzpatrick 2006) states that there exists a finite subcover $(B(\tau(x_i), \eta_{x_i}))_{i\in\{1,\dots,n\}}$ of $\tau([0,1])$. To this finite subcover corresponds a finite number of feasible trajectories, going from $\tau(0)$ to $\tau(1)$, included in the union of $(B(\tau(x_i), \epsilon_{x_i}))_{i\in\{1,\dots,n\}}$, and thus in $\mathcal{C}_{free}$. This concludes the proof. □

4.1.1 Small-Time *versus* Small-Space Controllability

In the control theory literature, the property used is usually *small-time controllability*, which states that for all configurations q, for all times $T > 0$, the set of configurations accessible from q in time less than T forms a neighborhood of q. When accelerations and velocities are bounded, small-time controllability implies small-space controllability. This is why a lot of motion planning previous work only refers to the sufficient small-time controllability property. However, the reciprocate is not necessarily true: a system can be small-space controllable and not small-time, if the trajectories generated by its controller are arbitrarily long. The important property, regarding motion planning application, is small-space controllability, as Theorem 1 shows. In the following, we show that legged robots are small-space controllable, but not that they are small-time controllable. In fact, the control method that we present does not follow the small-time controllability property. For the sake of clarity, we have chosen to make the distinction between these two controllability properties.

4.2 Dynamic Walking Makes Humanoid Robots Small-Space Controllable

This section discusses a walking robot small-space controllability. To clarify the presentation, we consider a simplified model of a legged robot consisting of two feet of zero mass and a point mass free to move in three dimensions. We do not consider the kinematic chains between the feet and the mass. The robot is walking on a flat terrain, and the feet are assumed to have a positive

surface. For our presentation, it is not necessary to consider the foot height, so the configuration space of the robot is:

$$\mathcal{C} = SE(2) \times SE(2) \times \mathbb{R}^3$$

It is of dimension 9.

The balanced walking conditions for a quasi-static walking robot are that the point mass, or CoM, should always be over the support polygon (the convex hull of the two feet), and one foot can move iff the CoM is over the other foot. Similarly, the walking conditions for a dynamic walking robot are that the ZMP should always be in the robot support polygon, and one foot can move iff the ZMP is over the other foot. For a precise description of dynamically balanced walking conditions, the reader can refer to (Wieber 2002). Under these assumptions, the following result holds:

Theorem 2. *A quasi-statically walking robot is not small-space controllable. A dynamically walking robot is.*

Proof of Theorem 2. The first claim is straightforward. Let the robot be in a configuration q where the two feet are separated by a positive distance. Let $L > 0$ be the positive horizontal distance between the CoM and the left foot (if the CoM is over the left foot, we can consider similarly the right foot). For all $\epsilon < L$, any valid trajectory starting from q, included in the ball of center q and radius ϵ, is such that the CoM is never over the left foot. Given the quasi-static walking conditions, the right foot of the robot is fixed along the trajectory. Thus, the set of accessible configurations from q by staying inside $B(q, \epsilon)$ does not form a neighborhood of q, since it does not include any configuration where the right foot has moved. This shows that the robot is not small-space controllable.

Let us now consider a dynamically walking robot. Starting from any valid static configuration, the CoM can move vertically without affecting balance, so there is no need to consider this degree of freedom in the following. If the CoM is not over the edge of the support polygon, it is possible to move it in a quasi-static way inside a neighborhood of its current position that projects itself over the support polygon. It is thus sufficient and necessary to prove that for all $\epsilon > 0$, it is possible to move the feet while keeping the CoM inside a neighborhood of size ϵ. Let such $\epsilon > 0$ be arbitrarily fixed.

The model of a walking robot with a point mass at a fixed height is known in the literature as the cart-table model (Kajita et al. 2003). The equations giving the ZMP horizontal coordinates (p_x, p_y) as functions of CoM horizontal coordinates (x, y) in the cart-table model were presented in (Kajita et al. 2003):

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} x - \frac{z_c}{g}\ddot{x} \\ y - \frac{z_c}{g}\ddot{y} \end{pmatrix} \tag{1}$$

where z_c is the constant height of the CoM and g is the gravity constant. In the following we will note $\omega_0 = \sqrt{\frac{g}{z_c}}$.

Without loss of generality, let us assume that the robot is in a configuration in which the CoM is at the horizontal position $(0, 0)$, the foot centers are aligned with the y-axis and the horizontal distance between the CoM and either of the foot centers is L. To achieve dynamically balanced walking, we aim at making $p_y(t)$ oscillate between $-L$ and L. To move the ZMP under a given foot, only the y coordinate of the CoM is of interest. Thus, we will keep the x coordinates of the CoM and ZMP constant equal to 0. By hypothesis, the feet have a positive surface, let $l > 0$ be such that the length of the section of a foot along the y-axis is greater than l. Fig. 9 summarizes the notations used in the following.

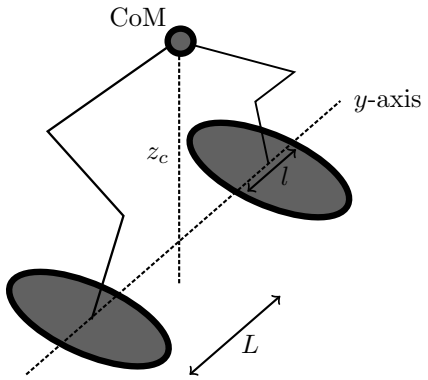

Figure 9: Simplified model of a legged robot. The CoM is at $(0, 0, z_c)$, the two feet are flat on the ground, aligned with the y-axis, at a horizontal distance L from the CoM.

The idea of this proof is to use the form of Eq. (1) to apply a scaling factor between the amplitude of the oscillations of the CoM and of the ZMP. The faster the CoM oscillates, the bigger is the amplitude of the ZMP oscillations. Following is a formalization of this idea.

For $\omega > 0$, assuming the CoM follows the trajectory $y(t) = \epsilon \sin(\omega t)$, Eq. (1) gives:

$$p_y(t) = (1 + \left(\frac{\omega}{\omega_0}\right)^2)\epsilon \sin(\omega t)$$

The amplitude of the oscillations of y is multiplied by a factor $(1 + \left(\frac{\omega}{\omega_0}\right)^2)$. Choosing $\omega = \omega_0\sqrt{\frac{L}{\epsilon} - 1}$ makes p_y oscillate between $-L$ and L, while y oscillates between $-\epsilon$ and ϵ. At time $t_l^{(n)} = n\frac{2\pi}{\omega} + \frac{\pi/2}{\omega}$, the ZMP is located at the center of the left foot, the robot can move its right foot and at time $t_r^{(n)} = n\frac{2\pi}{\omega} + \frac{3\pi/2}{\omega}$ the ZMP is located at the center of the right foot, the robot can move its left foot.

Starting from a static configuration at time $(t = 0)$, we cannot apply directly a command $y(t) = \epsilon \sin(\omega t)$ because it generates a discontinuity in the speed of the CoM at time $(t = 0)$. To overcome this discontinuity, we go through a

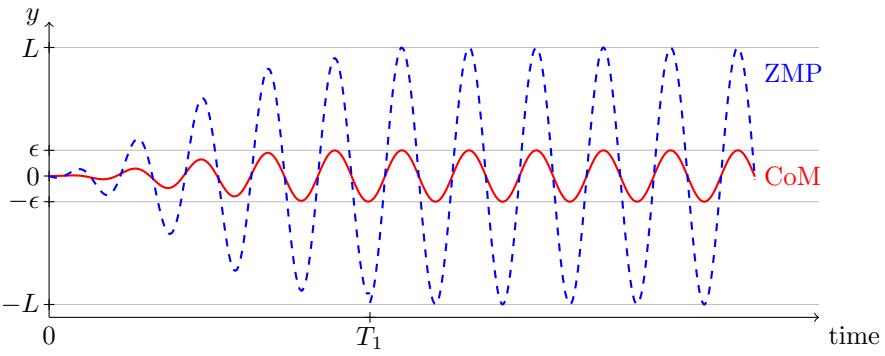

Figure 10: CoM motion (solid line) along y axis. The CoM stays in the interval $[-\epsilon, \epsilon]$ while during permanent state ($t \geq T$), the ZMP (dashed line) oscillates between the centers of the feet, which allows in-place walk.

transient state between ($t = 0$) and ($t = T$) for some $T > 0$. Let $f : [0, T] \to [0, 1]$ be an increasing function of class C^∞ such that $f(0) = 0$, $\dot{f}(0) = 0$, $f(T) = 1$, $\dot{f}(T) = 0$ and $\ddot{f}(T) = 0$. We can explicitly construct such an f with a spline of degree 4. We also request that for all $t \in [0, T]$, $|2\epsilon\dot{f}(t)\frac{\omega}{\omega_0^2}| \leq \frac{l}{4}$ and $|\epsilon\ddot{f}(t)/\omega_0^2| \leq \frac{l}{4}$. These inequalities will be used to bound the trajectory of the ZMP. We can guarantee them by choosing T large enough. Let us now consider the following CoM motion:

$$y(t) = \begin{cases} f(t)\epsilon\sin(\omega t) & \text{if } t \in [0, T] \\ \epsilon\sin(\omega t) & \text{if } t \geq T \end{cases}$$

One can check that y is of class C^2 over $\mathbb{R}_+$, and that $\dot{f}(0) = 0$. When $t \geq T$, the robot is in the permanent state described above and can successively move either of its feet inside small neighborhoods. The last point to check is that for $t \in [0, T]$ $p_y(t)$ stays inside the support polygon of the robot. The calculation of the successive derivatives of y gives:

$$\begin{aligned} p_y(t) = \quad & f(t)\epsilon(1 + \left(\tfrac{\omega}{\omega_0}\right)^2)\sin(\omega t) \\ & + 2\epsilon\dot{f}(t)\tfrac{\omega}{\omega_0^2}\cos(\omega t) \\ & + \tfrac{\epsilon}{\omega_0^2}\ddot{f}(t)\sin(\omega t) \end{aligned}$$

For all $t \in [0, T]$, $f(t)\epsilon(1 + \frac{\omega}{\omega_0}^2)\sin(\omega t)$ lies between $-L$ and L. The bounds on the derivatives of f guarantee that $p_y(t)$ lies between $-L - l/2$ and $L + l/2$, which means that the ZMP stays inside the support polygon. Fig. 10 shows an example of CoM motion on the y axis and the corresponding ZMP motion. Once in permanent in-place walking state, the robot can come back to a static state by applying a symmetric transient state used to decrease gradually the

amplitude of the oscillations of the CoM without generating a discontinuity in the first derivative of the command.

We have thus exhibited a continuous control scheme that allows to move any of the feet in any direction, while keeping the CoM inside an arbitrarily small neighborhood. This concludes the proof. □

4.2.1 Remarks

Generalization to a complete model: We did not extend the previous proof to any legged robot model since empirically, the table cart model is a very good fit for our humanoid robot. Although of little practical interest, the generalization of the proof does not seem very difficult to achieve. As an insight, the difference between the table cart model and the full size humanoid robot is due to the derivative of the angular momentum and to the vertical acceleration of the center of mass. These perturbations can be made as small as desired along the sliding path by following the sliding path as slowly as necessary. The derivatives of the angular momentum produced by the stepping motion can also be made as small as desired by making the step height as small as necessary and by using recent results on properties of joint trajectories induced by end-effector motions (Zanchettin & Rocco 2012).

Use of ZMP preview controller: The control strategy presented in the previous proof may generate very long trajectories, because of the transient states at the beginning and end of the locomotion. In the actual implementation, we have chosen to generate CoM motions with a ZMP preview controller, as presented in (Kajita et al. 2003). We have observed experimentally that the amplitude of CoM trajectories decreases when the frequency of steps increases. Our current ZMP preview controller relies on the cart-table model approximation. To make this approximation valid, we fix the height of the robot CoM during walk, as well as the vertical orientation of the robot waist. These geometric constraints are also applied when planning statically balanced paths, to ensure that the paths can be approximated by dynamic walk trajectories. Note that this is due to our current ZMP preview controller implementation, and does not affect the generality of the small-space controllability result presented above.

Relying on the cart-table model approximation means that the angular momentum induced by arm movements for instance can lead to non dynamically balanced walking motion. We thus implement the ZMP filtering stage proposed in (Kajita et al. 2003) to compute the exact ZMP, take into account the full dynamics of the robot and generate feasible trajectories.

Speed of CoM: The theoretical result presented in this section implies that any collision-free path can be approximated by a sequence of admissible and collision-free trajectories. However, the theorem depends on a control law that generates trajectories with unbounded velocities for the CoM, when the input path is close to obstacles. The humanoid robot hardware may be a limitation

to such trajectories. To prevent the generated CoM oscillations from being too fast, one has to require that the statically balanced path is included inside an ϵ-radius tube of the free space, where ϵ depends on the physical capacities of the robot.

4.3 Application: Dynamic Approximation of a Statically Balanced Sliding Path

The algorithm that animates a statically balanced path into a dynamically balanced walk trajectory has been inspired by the previous small-space controllability proof. Given a statically balanced path p verifying the cart-table model approximation constraints, we start by placing footprints corresponding to the nominal walk pattern of the robot. Given the footprints, we compute a ZMP trajectory, derive foot trajectories, and a preview controller returns the corresponding CoM trajectory. Classic numerical Jacobian-based prioritized inverse kinematics methods prove to be very useful to generate a dynamic walking trajectory while trying to accomplish secondary tasks, such as following a reference configuration trajectory. We use the framework called Generalized inverse kinematics (Gik) and developed in (Yoshida, Kanoun, Esteves & Laumond 2006).

The hierarchy of tasks (referred to as *GikTasks* in Algorithm 4) applied to the robot to generate a dynamic walking motion is – in decreasing priority order:

1. Positions and orientations of feet,
2. Horizontal position of the CoM,
3. Height of the CoM,
4. Verticality of the waist,
5. Configuration task towards corresponding configuration in p.

Tasks (1) and (2) generate a dynamically balanced motion by using the simplified cart-table model and the ZMP criterion. Tasks (3) and (4) ensure that the resulting motion is well described by the cart-table model. Task (5) is used to approximate p as well as possible given the walk parameters.

Because it comes at the lowest priority, task (5) is not necessarily fulfilled in the resulting trajectory. Hence, collisions may appear when animating p, if the resulting trajectory diverges too much from the initial sliding path. If so, it is necessary to approximate more closely p by a walk trajectory. To do so, we use the small-space controllability property of the system shown in the previous section. The way we use this property is inspired by similar results in non-holonomic mobile robot control presented in (Laumond et al. 1994).

If the animated trajectory collides with the environment, we cut the initial path p into two sub-paths, that we try to animate recursively. When the paths to animate are too short for the robot nominal walk parameters, we accelerate the steps, and decrease the maximum height of the moving foot. As shown in previous section, the walk trajectory corresponding to smaller and faster steps

converges toward the sliding path. Algorithm 4 shows pseudo-code that takes a sliding path p as input and returns a collision-free walk trajectory[2].

Algorithm 4 FindDynamicTrajectory(Path p)

```
Footprints ← ComputeFootprints(p)
GikTasks.addFootprintTask(Footprints)
GikTasks.addWaistTask()
GikTasks.addConfigurationTask(p)
DynamicTrajectory ← ComputeWalkTrajectory(GikTasks)
if (CheckForCollisions(DynamicTrajectory) = Colliding) then
  (p1, p2) ← CutInHalf(p)
  DT1 ← FindDynamicTrajectory(p1)
  DT2 ← FindDynamicTrajectory(p2)
  return Concatenate(DT1, DT2)
else
  return DynamicTrajectory
end if
```

5 Experimental Results

The motion planning algorithms presented in this paper have been implemented using KineoWorks^{TM}(Laumond 2006). The planning times have been measured on an Intel Core 2 Duo 2.13 GHz PC with 2 GB of RAM. Evaluation of the randomized algorithm has been conducted by executing 500 trials on each scenario using two flavors of RRT: the classic RRT and IPP-RRT (Ferre & Laumond 2004). We present the results in Appendix A, Fig. 17, 18, 19, as well as in Extension 4 for the raw data.

Our whole-body motion planner generates a robot configuration trajectory that is sampled at a 200 Hz rate and stored in a file. This file can then be used to play the trajectory in open-loop on the HRP-2 robot, which is position-controlled. Scenarios in Sections 5.1 and 5.3 were both successfully executed.

To get more natural motions in the experiments, we require the foot positions to be fixed with respect to each other, and the CoM to be projected in the center of the support polygon during the sliding path planning stage.

5.1 Passing between two chairs

The environment shown in Fig. 1 and 6 was presented in (El Khoury, Taix & Lamiraux 2011). There, the motion planning problem is solved with a bounding box method, leading the robot to walk sideways between the two chairs. Our method generates a locomotion trajectory in which the robot walks forward, which may be required if the robot has to use vision during locomotion. The

[2]The actual implementation of this algorithm is part of an open-source package, available at `https://github.com/laas/hpp-wholebody-step-planner`.

first planning stage requires 1 s on average. The animation of the sliding path presented in Fig. 6 uses 66.5 s of computation time.

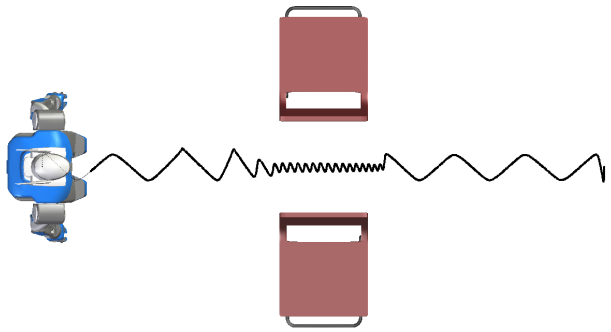

Figure 11: Horizontal trajectory of the robot CoM during locomotion. When the robot is close to obstacles, the amplitude of the oscillations decreases.

Fig. 11 shows the horizontal trajectory of the robot CoM during locomotion. The amplitude of the oscillations decreases when passing between the chairs. This motion has been validated on a real HRP-2 platform. Extension 2 shows a video of the experiment.

5.2 Walking among floating obstacles

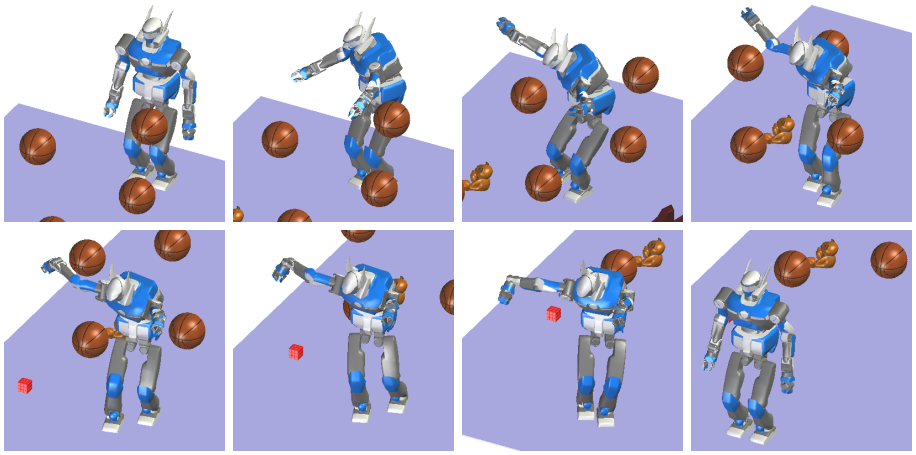

Figure 12: Solution path for a cluttered environment, the robot walks among floating obstacles.

In the environment shown in Fig. 12, the robot has to find a way among floating obstacles. In this environment neither bounding box nor footstep plan-

ning strategies could find a collision-free walk trajectory. The first planning stage requires 53 s on average, and the animation of the trajectory presented in Fig. 12 uses 339.5 s of computation time. Fig. 13 shows the robot CoM trajectory during locomotion.

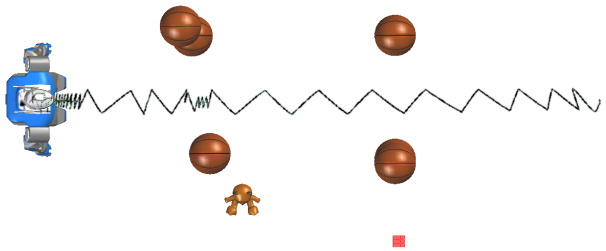

Figure 13: Horizontal trajectory of the robot CoM during locomotion.

5.3 'Put the ball on a shelf'

In the problem shown in Fig. 14 the robot has to put a ball on a shelf, in a constrained apartment environment. The final configuration is defined implicitly as a desired hand position. We have generated automatically goal configurations solving the task, as described in Section 3 . Then, we have applied our planner to generate a whole-body walking motion that solves the hand reaching task.

The solution sliding path is constrained between the table on the right and the lamp on the left. This passage is too narrow for the robot nominal walk parameters. When executing the walk motion resulting from our algorithm, the robot left hand is only a few centimeters away from the lamp.

The first planning stage requires 15 s on average, and the animation of the resulting walk motion presented in Fig. 14 requires around 190 s of computation time. Fig. 15 shows the robot CoM trajectory during locomotion. Extension 3 presents a comprehensive video of this problem, including the motion execution on the real robot HRP-2 on stage. Fig. 16 shows some snapshots taken from Extension 3.

6 Limits and Discussions

This section lists several limitations of the current methods, and discusses potential future work to overcome them.

6.1 Stepping over obstacles

Because of the kinematic constraints we apply at the planning stage, we are not able yet to plan motions where the robot steps over obstacles, while this is an important feature of humanoid robots. Nevertheless, because we compute

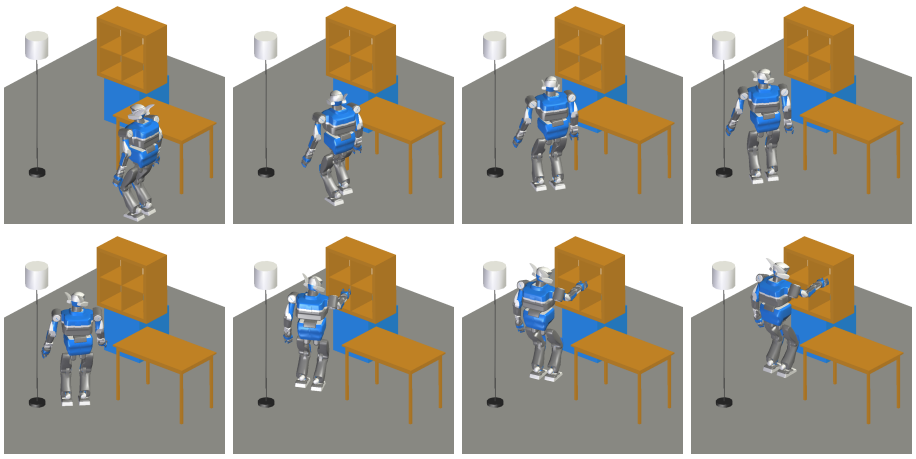

Figure 14: Solution path for a hand reaching problem in an apartment. The goal is implicitly defined as an inverse kinematics task.

collision queries on an exact model of the robot, our method is able to generate paths where obstacles pass between the feet of the robot. In future work, we plan to develop mixed methods, where collision avoidance at the leg level can be solved by footstep planning techniques, while whole-body collision-avoidance can be solved by the algorithm presented in this paper.

6.2 Environment representation

The experimental setup assumes perfect knowledge of the environment. This can be guaranteed during experiments by using calibrated objects and motion capture systems. This indeed allows us to focus on complex motion planning problems. The perception problem, interesting as it is, is thus completely decoupled from the planning problem in our presentation. From previous experiences however, we are confident that our algorithm will work as well in environments modeled by vision (Nakhaei & Lamiraux 2008, Dang, Lamiraux & Laumond 2012).

6.3 Trajectory following

The setup also assumes perfect execution of the plan. It can be critical here, since non-nominal stepping may cause the robot to drift away from the planned trajectory. Future experiments will include trajectory following during plan execution.

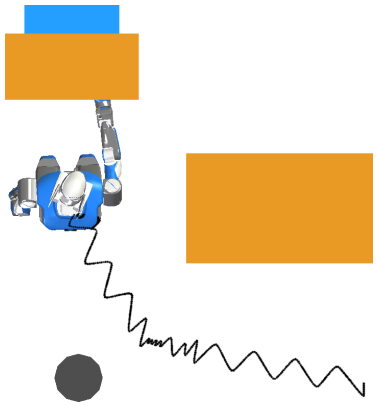

Figure 15: Horizontal trajectory of the robot CoM during locomotion.

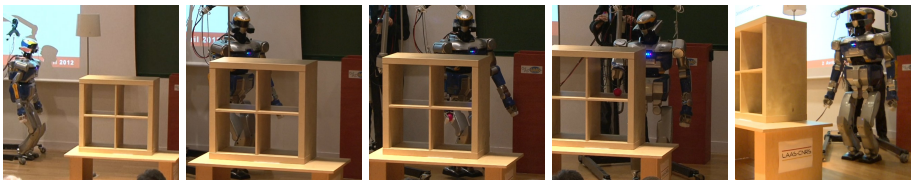

Figure 16: Execution of the walking trajectory by HRP-2 on stage. The robot first goes to the shelves to release the ball, then comes back to a rest position.

7 Conclusion

In this paper, we have presented a simple algorithm for constrained motion planning and used it within a novel, well-grounded strategy for humanoid whole-body manipulation planning including locomotion. The locomotion algorithm is based on a formal small-space controllability property of humanoid robots. An important point is that this strategy only holds for dynamic walking robots, and not for quasi-static walking ones. We have used our motion planner on different challenging examples, and validated the generated motions on a real platform. We have discussed the limits and potential extensions of our method, and we plan to address them in future work.

8 Acknowledgments

This work was supported by the French FUI Project ROMEO and the European Project ECHORD-231143. The authors would like to thank Thomas Moulard and Olivier Stasse for their valuable help during experiments.

A Sliding Motion Planning Benchmarks

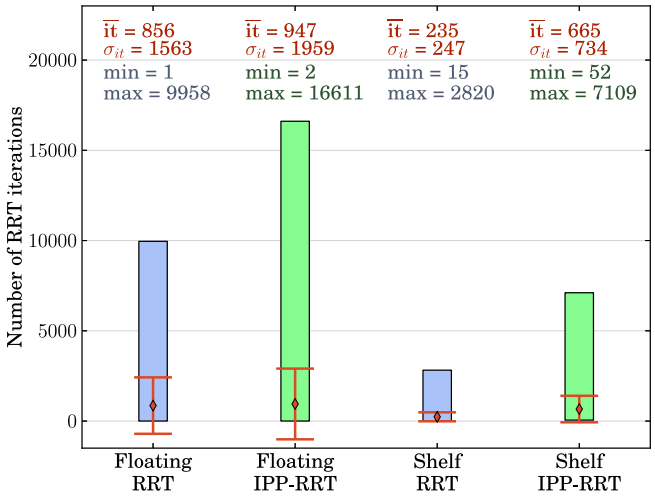

Figure 17: Number of RRT iterations it for the floating objects and the shelf scenarios, using two variants of RRT. Mean $\overline{it}$, standard deviation σ_{it}, minimum and maximum values are represented.

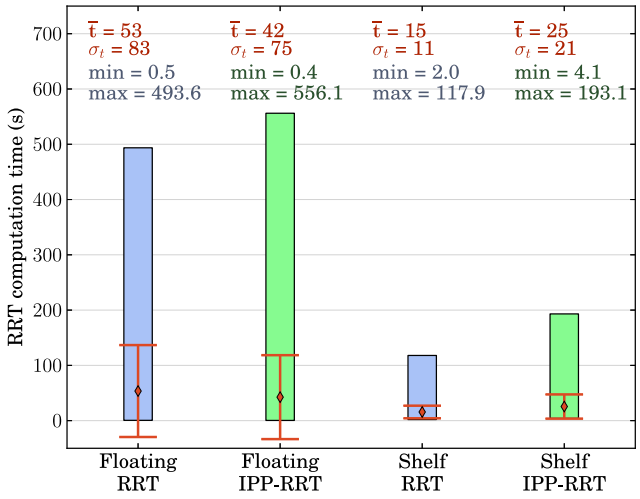

Figure 18: RRT computation time t for the floating objects and the shelf scenarios, using two variants of RRT. Mean $\overline{t}$, standard deviation σ_t, minimum and maximum values are represented.

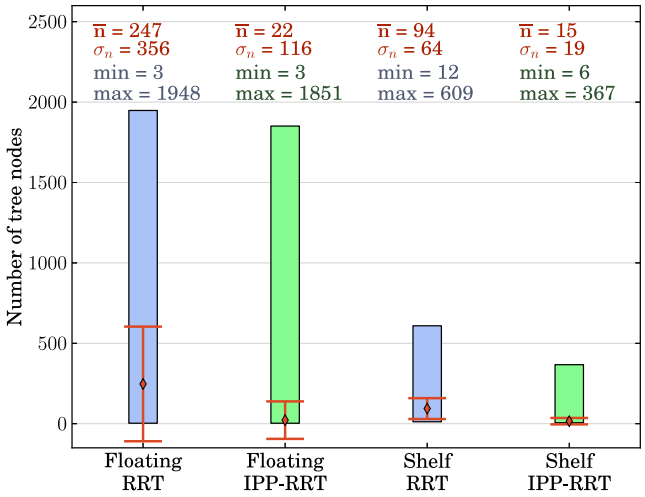

Figure 19: Number of tree nodes n for the floating objects and the shelf scenarios, using two variants of RRT. Mean $\overline{n}$, standard deviation σ_n, minimum and maximum values are represented.

B Index to Multimedia Extensions

Extension	Type	Description
1	Video	Example of a constrained motion planning result executed on HRP-2.
2	Video	Chairs scenario: solution of whole-body planning executed on HRP-2.
3	Video	Shelf scenario: solution of whole-body planning executed in simulation and on HRP-2.
4	Data	Raw data of sliding motion planning benchmarks for all scenarios.

References

Baerlocher, P. & Boulic, R. (1998), Task-priority formulations for the kinematic control of highly redundant articulated structures, *in* '1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998. Proceedings.', Vol. 1.

Berenson, D., Srinivasa, S. & Kuffner, J. (2011*a*), 'Task Space Regions: A framework for pose-constrained manipulation planning', *The International Journal of Robotics Research* .

Berenson, D., Srinivasa, S. S. & Kuffner, J. (2011*b*), 'Task space regions: A framework for pose-constrained manipulation planning', *The International Journal of Robotics Research* .

Bonnans, J., Gilbert, J., Lemaréchal, C. & Sagastizábal, C. (2006), *Numerical optimization: theoretical and practical aspects*, Springer.

Bretl, T. (2006), 'Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem', *The International Journal of Robotics Research* **25**(4), 317–342.

Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J. & Kanade, T. (2005), Footstep planning for the Honda ASIMO humanoid, *in* 'Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on', IEEE, pp. 629–634.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. & Thrun, S. (2005), *Principles of Robot Motion: theory, algorithms, and implementation*, MIT Press.

Dalibard, S., El Khoury, A., Lamiraux, F., Taix, M. & Laumond, J. (2011), Small-space controllability of a walking humanoid robot, *in* 'Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on', IEEE, pp. 739–744.

Dalibard, S., Nakhaei, A., Lamiraux, F. & Laumond, J.-P. (2009), Whole-body task planning for a humanoid robot: a way to integrate collision avoidance, *in* 'Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on', pp. 355 –360.

Dang, N., Lamiraux, F. & Laumond, J.-P. (2012), Experiments on whole-body manipulation and locomotion with footstep real-time optimization, *in* 'IEEE International Conference on Humanoid Robots (Humanoids)', Osaka.

Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S. & Kuffner, J. (2008), 'Bispace planning: Concurrent multi-space exploration', *Proceedings of Robotics: Science and Systems IV* .

El Khoury, A., Taix, M. & Lamiraux, F. (2011), 'Path optimization for humanoid walk planning: an efficient approach', *Int. Conf. Informatics in Control, Automation and Robotics (ICINCO) 2011* .

Ferre, E. & Laumond, J. (2004), An iterative diffusion algorithm for part disassembly, *in* '2004 International Conference on Robotics and Automation (ICRA'2004)', New Orleans (USA), pp. 3149–3154.

Fitzpatrick, P. (2006), *Advanced calculus*, Vol. 5, Amer Mathematical Society.

Hsu, D., Latombe, J. & Motwani, R. (1999), 'Path planning in expansive configuration spaces', *Int. J. Computational Geometry & Applications* **9**(4-5), 495–512.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K. & Hirukawa, H. (2003), Biped walking pattern generation by using preview control of zero-moment point, *in* 'IEEE International Conference on Robotics and Automation', Vol. 2, Citeseer, pp. 1620–1626.

Kanehiro, F., Lamiraux, F., Kanoun, O., Yoshida, E. & Laumond, J. (2008), A Local Collision Avoidance Method for Non-strictly Convex Polyhedra, *in* '2008 Robotics: Science and Systems Conference'.

Karaman, S. & Frazzoli, E. (2011), 'Sampling-based algorithms for optimal motion planning', *The International Journal of Robotics Research* **30**(7), 846–894.
URL: *http://ijr.sagepub.com/content/30/7/846.abstract*

Kavraki, L., Svestka, P., Latombe, J. & Overmars, M. (1996), 'Probabilistic roadmaps for path planning in high-dimensional configuration spaces', *Robotics and Automation, IEEE Transactions on* **12**(4), 566–580.

Khatib, O., Sentis, L., Park, J. & Warren, J. (2004), 'Whole body dynamic behavior and control of human-like robots', *International Journal of Humanoid Robotics* **1**(1), 29–43.

Kuffner, J. & LaValle, S. (2000), RRT-connect: An efficient approach to single-query path planning, *in* 'Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA'.
URL: *citeseer.ist.psu.edu/article/kuffner00rrtconnect.html*

Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M. & Inoue, H. (2005), 'Motion planning for humanoid robots', *Robotics Research* pp. 365–374.

Kuffner Jr, J., Nishiwaki, K., Kagami, S., Inaba, M. & Inoue, H. (2001), Footstep planning among obstacles for biped robots, *in* 'Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on', Vol. 1, IEEE, pp. 500–505.

Laumond, J. (2006), 'Kineo CAM: a success story of motion planning algorithms', *Robotics & Automation Magazine, IEEE* **13**(2), 90–93.

Laumond, J.-P., Jacobs, P., Taix, M. & Murray, R. (1994), 'A motion planner for nonholonomic mobile robots', *Robotics and Automation, IEEE Transactions on* **10**(5), 577 –593.

LaValle, S. M. (2006), *Planning Algorithms*, Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

Nakamura, Y. & Hanafusa, H. (1986), 'Inverse kinematic solutions with singularity robustness for robot manipulator control', *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control* **108**, 163–171.

Nakhaei, A. & Lamiraux, F. (2008), Motion planning for humanoid robots in environments modeled by vision, *in* 'Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on', pp. 197 –204.

Perrin, N., Stasse, O., Baudouin, L., Lamiraux, F. & Yoshida, E. (2012), 'Fast humanoid robot collision-free footstep planning using swept volume approximations', *Robotics, IEEE Transactions on* **28**(2), 427 –439.

Pettré, J., Laumond, J. & Siméon, T. (2003), A 2-stages locomotion planner for digital actors, *in* 'Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation', Eurographics Association, p. 264.

Porta, J., Jaillet, L. & Bohigas, O. (2012), 'Randomized path planning on manifolds based on higher-dimensional continuation', *The International Journal of Robotics Research* **31**(2), 201–215.

Shkolnik, A., Levashov, M., Manchester, I. & Tedrake, R. (2011), 'Bounding on rough terrain with the littledog robot', *The International Journal of Robotics Research* **30**(2), 192.

Siciliano, B. & Slotine, J. (1991), A general framework for managing multiple tasks in highly redundant robotic systems, *in* 'Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on', pp. 1211–1216.

Toussaint, M., Gienger, M. & Goerick, C. (2007), Optimization of sequential attractor-based movement for compact behaviour generation, *in* 'IEEE International Conference on Humanoid Robots', pp. 122–129.

Vukobratovic, M. & Juricic, D. (1969), 'Contribution to the synthesis of biped gait', *Biomedical Engineering, IEEE Transactions on* (1), 1–6.

Wieber, P.-B. (2002), On the stability of walking systems, *in* 'Proceedings of the International Workshop on Humanoid and Human Friendly Robotics', Tsukuba, Japan.
URL: *http://hal.inria.fr/inria-00390866/en*

Yoshida, E., Belousov, I., Esteves, C. & Laumond, J.-P. (2005), Humanoid motion planning for dynamic tasks, *in* 'Humanoid Robots, 2005 5th IEEE-RAS International Conference on', pp. 1 –6.

Yoshida, E., Kanoun, O., Esteves, C. & Laumond, J. (2006), Task-driven support polygon reshaping for humanoids, *in* 'Humanoid Robots, 2006 6th IEEE-RAS International Conference on', pp. 208–213.

Zanchettin, A. & Rocco, P. (2012), 'A general user-oriented framework for holonomic redundancy resolution in robotic manipulators using task augmentation', *Robotics, IEEE Transactions on* **28**(2), 514 –521.