

Multi-Contact Stances Planning for Multiple Agents

Karim Bouyarmane and Abderrahmane Kheddar

Abstract—We propose a generalized framework together with an algorithm to plan a discrete sequence of multi-contact stances that brings a set of collaborating robots and manipulated objects from a specified initial configuration to a desired goal through non-gaited acyclic contacts with their environment or among each other. The broad range of applications of this generic algorithm includes legged locomotion planning, whole-body manipulation planning, dexterous manipulation planning, as well as any multi-contact-based motion planning problem that might combine several of these sub-problems. We demonstrate the versatility of our planner through example scenarios taken from the aforementioned classes of problems in virtual environments.

I. INTRODUCTION

Recent works [1], [2] started tackling the acyclic motion planning problem for humanoid and/or legged robots taking a *contacts-before-motion* planning approach. The approach is based on planning a feasible sequence of stances from an initial configuration to a goal configuration, before planning the subsequent continuous motion that goes through this planned sequence of stances. This paper is concerned only with the first part of the problem, i.e. the discrete stances sequence planning sub-problem. Such a decoupling scheme of the two components of the problem, though less theoretically founded in terms of completeness than the interleaved approach of *multi-modal planning* [3], enables us to reduce the complexity of the problem and yet still manages to solve highly constrained situations as demonstrated on practical real-life humanoid robot experiments [4], [5], [6].

The core algorithm we are using here was first introduced in the works of Escande et al. [4]. In its most reduced form, it is a *Best-First Planning* (BFP) algorithm [7], [8] that explores the continuum of the workspace for finding best contact locations, as opposed to the main other method first introduced in the works of Hauser et al. [9] requiring prior discretization of possible contact locations on the environment. A major drawback of this latter approach resides in the difficult trade-off between the possible combinatorial issues that would be raised by too many pre-discretized locations, versus the possible misses of solutions induced by too few pre-discretized locations.

In this paper we build on this BFP-based algorithm and propose a novel framework that makes it possible, with one unique planner, to solve different classes of robotics contacts planning problems, beyond the initially targeted “legged locomotion for a single robot” problem. Such a planner can

solve, for example, the non-gaited dexterous manipulation problem, some example approaches of which can be found in the past few years’ literature [10], [11], [12], [13]. A more original contribution is to solve the contacts planning problem for collaborative robots manipulating objects [14]. The needed synchronization of contacts planning for the cooperative carrying of a heavy object by two humanoid robots is one example of the results of the planner. Additionally, by unifying locomotion and manipulation, the planner can also solve contact planning problems for situations interleaving both, which can prove useful for platforms such as humanoid robots that are designed to execute both locomotion and manipulation tasks.

These contributions (extension to multiple agents, generalization to any robotic platform, and non-decoupling of locomotion and manipulation) are made possible thanks to a formulation of the problem that reaches a higher level of abstraction, necessary in order to achieve the desired generalization. It allows us to make the extensions listed above with little rewriting effort of the existing algorithms. In other words, the algorithms here are the same as their original form [1], [2]; by generalizing the formalism and the framework, we extend their capabilities to a wider range of applications. This is our main contribution.

However, we emphasize once again that addressing the continuous motion generation problem is beyond the scope of this paper. In our previous approaches [4], [5], [6], the stances planning and continuous motion planning are two independent stages of a global planning framework. They are addressed as independent problems. This decoupled approach is justified by the experimental validation presented in our previous works [4], [5], [6]. Moreover, a quantitative analysis of completeness and global optimality issues of the proposed algorithms is also beyond the scope of the paper. Since no fundamental algorithmic contributions are brought along with our generalization process, these algorithmic issues/characteristics are no different from the ones encountered in the original works [1], [2] to which we refer the interested reader.

The rest of this paper is organized as follows. We first propose a formulation of the problem using the language and formalism of basic set theory (Section II). We then write our algorithm in this synthetic language and compare it with the other existing method (Section III). Last we demonstrate some results obtained by our planner on different classes of problems (Section IV).

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

II. PRELIMINARY NOTATIONS AND DEFINITIONS

In this section we will introduce the set-theoretic formalism that will make the extensions and the locomotion-manipulation unification process easier to write. The abstraction effort invested in this section will later be rewarded in the algorithms writing section (Section III). It will allow us to write the algorithms in a very generic, synthetic, and rigorous style. It might be useful to recall beforehand that, within this formalism, for any two sets A and B , $p : A \rightarrow B$ denotes a mapping from A to B , $\mathcal{P}(A)$ denotes the power set of A (set of subsets of A), $\text{card}(A)$ the cardinality of A , and for any two subsets $A' \in \mathcal{P}(A)$ and $B' \in \mathcal{P}(B)$, $p(A')$ and $p^{-1}(B')$ denote respectively the direct and inverse images of A' and B' under the mapping p . We use the symbol $A_1 \setminus A_2$ to denote the difference of two subsets $A_1, A_2 \in \mathcal{P}(A)$.

So let us suppose we have a system of N robots indexed in the set $\{1, \dots, N\}$. A “robot” here is either a fully- or under-actuated articulated mechanism or a non-actuated manipulated object. The environment can also be considered as a special case of “robot”, indexed with 0. Thus the index set $\{0, \dots, N\}$ contains all the articulated mechanisms, the manipulated objects, and the environment.

Each robot $r \in \{0, \dots, N\}$ can be represented as a kinematic tree made of b_r bodies (nodes of the tree) indexed in $\{0, \dots, b_r - 1\}$, linked by j_r actuated joints (edges of the tree) indexed in $\{1, \dots, j_r\}$ (or \emptyset if $j_r = 0$). See Fig. 1.

- $b_r = 1$ and $j_r = 0$ if r refers to the environment or to a manipulated object.
- The index 0 in the set $\{0, \dots, b_r - 1\}$ refers to the root body of the kinematic tree representing the robot r .

The configuration q of the system is an element of $\mathcal{C} = \prod_{r=1}^N \mathcal{C}_r$, the Cartesian product of the configuration spaces of the individual robots of the system. Hence

$$q = (q_1, \dots, q_N),$$

where, for $r \in \{1, \dots, N\}$,

- $q_r = (x_r, y_r, z_r, \alpha_r, \beta_r, \gamma_r, \delta_r, \theta_{r,1}, \dots, \theta_{r,j_r})$, if r refers to a free-base articulated mechanism such as a humanoid robot for example, the first seven components representing the 3D position and the unit quaternion-parametrized orientation of its root body indexed by 0.
- $q_r = (x_r, y_r, z_r, \alpha_r, \beta_r, \gamma_r, \delta_r)$, if r refers to a rigid non-articulated manipulated object.
- $q_r = (\theta_{r,1}, \dots, \theta_{r,j_r})$, if r refers to a fixed-base manipulator such as the finger of a multi-fingered dexterous hand for example.
- q_r is not defined for $r = 0$ (the environment). It could be if we were considering deformable environment for example.

On each robot $r \in \{0, \dots, N\}$ we further specify a set of m_r planar surface patches indexed in $\{1, \dots, m_r\}$. A pair $(r, s) \in \{0, \dots, N\} \times \{1, \dots, m_r\}$, which characterizes the surface, refers to an element $(b'_{r,s}, T_{r,s})$ of $\{0, \dots, b_r - 1\} \times SE(3)$, where $b'_{r,s}$ denotes the body to which the surface (r, s) is rigidly attached and $T_{r,s} = (o_{r,s}, \vec{x}_{r,s}, \vec{y}_{r,s}, \vec{z}_{r,s})$ denotes a frame attached to the body $b'_{r,s}$, such that the

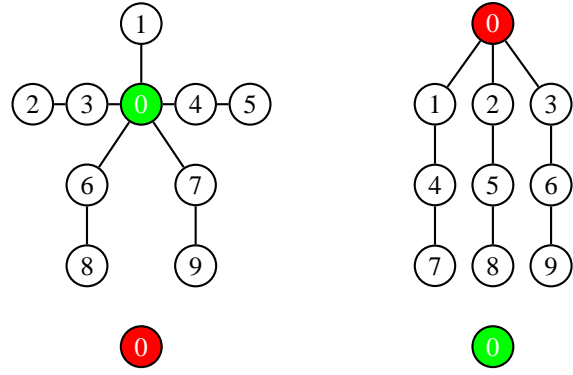


Fig. 1. Examples of the 4 types of kinematic trees yielding configuration space. Top left: a humanoid robot. Top right: a dexterous hand. Bottom left: the environment. Bottom right: a manipulated object. In red: fixed base. In green: free-flying base. A system of robots consists of an arbitrary number of any of those 4 types of kinematics trees.

point $o_{r,s}$ belongs to the surface, the vector $\vec{z}_{r,s}$ is the inwards normal to the surface, and the vectors $\vec{x}_{r,s}, \vec{y}_{r,s}$ are tangential to the surface. More general (i.e. non-planar) surface patches can be handled by considering *normalized Gauss frames* [15].

A contact is given by the specification of the two surfaces in contact (r_1, s_1) and (r_2, s_2) (i.e. a 4-tuple (r_1, s_1, r_2, s_2)) as well as their relative position/orientation (x, y, θ) . More precision is found in the following definition:

Definition 1 (contact, set of contacts E): A contact is a 7-tuple $(r_1, s_1, r_2, s_2, x, y, \theta)$, such that $r_1 \in \{1, \dots, N\}$, $r_2 \in \{0, \dots, N\}$, $r_2 \leq r_1$, $s_1 \in \{1, \dots, m_{r_1}\}$, $s_2 \in \{1, \dots, m_{r_2}\}$, $s_2 < s_1$ if $r_1 = r_2$, $b'_{r_1, s_1} \neq b'_{r_2, s_2}$ if $r_1 = r_2$, and $(x, y, \theta) \in \mathbb{R}^2 \times \mathbb{S}^1$. We define the *set of contacts E* as the subset of $\mathbb{N}^4 \times \mathbb{R}^2 \times \mathbb{S}^1$ made of such 7-tuples.

Remark 1: We can notice that this very generic definition only excludes environment-environment contacts ($r_1 \neq 0$), all other contact situations are possible, including a contact between two different bodies of the same robot (case $r_1 = r_2$). The ordering imposed on (r_1, r_2) and on (s_1, s_2) if $r_1 = r_2$ is required to avoid representing twice the same contact situation.

A contact $(r_1, s_1, r_2, s_2, x, y, \theta)$ geometrically corresponds to setting

$$\vec{z}_{r_1, s_1}(q) = -\vec{z}_{r_2, s_2}(q), \quad (1)$$

$$\vec{x}_{r_1, s_1}(q) = \cos(\theta)\vec{x}_{r_2, s_2}(q) + \sin(\theta)\vec{y}_{r_2, s_2}(q), \quad (2)$$

$$\vec{y}_{r_1, s_1}(q) = \sin(\theta)\vec{x}_{r_2, s_2}(q) - \cos(\theta)\vec{y}_{r_2, s_2}(q), \quad (3)$$

$$o_{r_1, s_1}(q) = o_{r_2, s_2}(q) + x\vec{x}_{r_2, s_2}(q) + y\vec{y}_{r_2, s_2}(q). \quad (4)$$

We call these equations the *contact equations*. They are illustrated in Fig. 2. Once again, for simplicity these are restricted to the planar surfaces case; for surfaces modeled as manifolds, the more general contact equations [15] should be considered (see Section IV for our practical handling of non-planar surfaces).

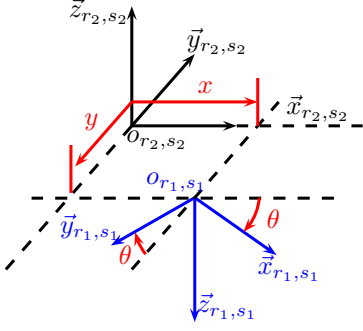


Fig. 2. Geometric illustration of a contact $(r_1, s_1, r_2, s_2, x, y, \theta)$.

On $\mathbb{N}^4 \times \mathbb{R}^2 \times \mathbb{S}^1$ we consider the projection map $p_{\mathbb{N}^4} : (r_1, s_1, r_2, s_2, x, y, \theta) \mapsto (r_1, s_1, r_2, s_2)$ which keeps only the first 4 components of the 7-tuple. $p_{\mathbb{N}^4}$ maps a contact to the pair of surfaces that constitute that contact.

Definition 2 (stance, set of stances Σ): A *stance* σ is a subset of the set of contacts E such that every pair of surfaces appears at most once. The set of all stances is denoted Σ ,

$$\Sigma = \left\{ \sigma \in \mathcal{P}(E) \mid \forall c_1, c_2 \in \sigma : c_1 \neq c_2 \Rightarrow p_{\mathbb{N}^4}(c_1) \neq p_{\mathbb{N}^4}(c_2) \right\}.$$

Remark 2: A stance σ is necessarily a finite subset of E , given that

$$\text{card}(\sigma) \leq \text{card}(p_{\mathbb{N}^4}(E)) \leq N(N+1) \left(\max_{r \in \{0, \dots, N\}} m_r \right)^2.$$

Every configuration of the system of robots defines a unique stance made of all the contacts for the robots in that configuration. Let us then denote $p_{\Sigma} : \mathcal{C} \rightarrow \Sigma$ the “forward kinematics” mapping that maps every configuration q to its stance σ . Inversely, each stance σ defines an “inverse kinematics” submanifold \mathcal{Q}_{σ} of the configuration space containing all the configurations that satisfy the contact equations (1), (2), (3), and (4) for all the contacts in the stance,

$$\mathcal{Q}_{\sigma} = p_{\Sigma}^{-1}(\{\sigma\}).$$

On this submanifold we isolate a special subspace of same dimensionality but less volume \mathcal{F}_{σ} in which the configurations are physically valid static configurations (i.e. configurations that are in static equilibrium, collision-free, for which the joint angles and torques are within their prescribed bounds).

The planning we will perform will be made on the set of stances Σ , rather than on the configuration space \mathcal{C} as it is the case in usual motion planning. We thus need to define an adjacency relation between stances. Two stances will be considered adjacent if they differ by exactly one contact. To formalize this we define the binary relation “have one contact less than”, that we denote \sqsubset , as

$$\sigma_1 \sqsubset \sigma_2 \quad \text{if} \quad \sigma_1 \subset \sigma_2 \quad \text{and} \quad \text{card}(\sigma_2) = \text{card}(\sigma_1) + 1.$$

Definition 3 (adjacency): Two stances σ_1 and σ_2 are said to be *adjacent* if $\sigma_1 \sqsubset \sigma_2$ or $\sigma_2 \sqsubset \sigma_1$. Given a stance σ we define the three following *adjacency sets*: $\text{Adj}^+(\sigma)$ the set of stances that add one contact to σ , $\text{Adj}^-(\sigma)$ the set of stances that remove one contact from σ , and $\text{Adj}(\sigma)$ the set of stances that are adjacent to σ (add or remove one contact). Formally:

$$\begin{aligned} \text{Adj}^+(\sigma) &= \{ \sigma' \in \Sigma \mid \sigma \sqsubset \sigma' \}, \\ \text{Adj}^-(\sigma) &= \{ \sigma' \in \Sigma \mid \sigma' \sqsubset \sigma \}, \\ \text{Adj}(\sigma) &= \text{Adj}^+(\sigma) \cup \text{Adj}^-(\sigma). \end{aligned}$$

A step in the plan will be a transition from one stance to an adjacent stance. Such a step will be feasible if there exists a common transition configuration that realizes both stances at the same time, i.e. if the intersection of the respective feasible spaces of the two stances is non-empty. This motivates the following definition:

Definition 4 (feasible sequence of stances): A sequence of stances $(\sigma_1, \dots, \sigma_n) \in \Sigma^n$, $n \geq 2$, is said to be *feasible* if it is made of a succession of adjacent stances with common transition configurations between two successive stances

$$\forall i \in \{1, \dots, n-1\} \quad \sigma_{i+1} \in \text{Adj}(\sigma_i) \quad \text{and} \quad \mathcal{F}_{\sigma_i} \cap \mathcal{F}_{\sigma_{i+1}} \neq \emptyset.$$

We can now formulate the problem we want to solve:

Problem 1 (non-gaited stances planning problem):

Given two stances σ_{start} and σ_{goal} in Σ , find a feasible sequence of stances $(\sigma_1, \dots, \sigma_n)$ such that $\sigma_1 = \sigma_{\text{start}}$ and $\sigma_n = \sigma_{\text{goal}}$.

The ability to solve Problem 1 rather than cyclic gaited steps planning problems makes the robots more autonomous in handling unexpectedly structured environment. Note, however, that in many simple cases, gaited sequences emerge automatically (“naturally”) in our results from solving Problem 1 (cf. Section IV).

Remark 3: We can also specify the goal to reach in terms of a configuration q_{goal} rather than a stance σ_{goal} . In this case we get the same formulation as Problem 1 where σ_{goal} simply denotes $p_{\Sigma}(q_{\text{goal}})$. These are actually the kind of queries we are addressing in Section IV.

Solving Problem 1 in a greedy algorithmic way amounts to exploring $\text{Adj}(\sigma)$ for a given σ , choosing $\sigma' \in \text{Adj}(\sigma)$, finding a configuration q in $\mathcal{F}_{\sigma} \cap \mathcal{F}_{\sigma'}$ to validate the choice of σ' , and iterating on σ' . Let us then analyse more closely the structure of $\text{Adj}(\sigma)$ for a given $\sigma \in \Sigma$. First, we should rewrite constructive expressions of the adjacency sets. From Definition 3 it follows that

$$\begin{aligned} \text{Adj}^+(\sigma) &= \left\{ \sigma \cup \{c\} \mid c \in p_{\mathbb{N}^4}^{-1} \left(p_{\mathbb{N}^4}(E) \setminus p_{\mathbb{N}^4}(\sigma) \right) \right\}, \\ \text{Adj}^-(\sigma) &= \left\{ \sigma \setminus \{c\} \mid c \in \sigma \right\}. \end{aligned}$$

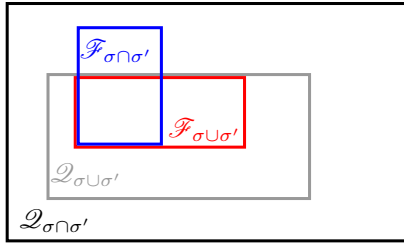


Fig. 3. Venn diagrams illustrating Proposition 1.

The removing-one-contact set $\text{Adj}^-(\sigma)$ is thereby a finite set, with $\text{card}(\text{Adj}^-(\sigma)) = \text{card}(\sigma)$. The adding-one-contact set $\text{Adj}^+(\sigma)$, however, needs to be more finely structured. When adding a contact $(r_1, s_1, r_2, s_2, x, y, \theta)$, we first choose the two surfaces (r_1, s_1) and (r_2, s_2) that we want to add as a contact, then we decide what their relative position/orientation (x, y, θ) will be. A nice way to formalize this is through equivalence classes. Let us define on $\text{Adj}^+(\sigma)$ the following equivalence relation

$$\begin{aligned} \sigma_1 \sim_\sigma \sigma_2 \text{ if} \\ \sigma_1 = \sigma \cup \{c_1\} \text{ and } \sigma_2 = \sigma \cup \{c_2\} \text{ and } p_{\mathbb{N}^4}(c_1) = p_{\mathbb{N}^4}(c_2). \end{aligned}$$

This equivalence relation only makes distinction between the surface pairs in the added contacts with no consideration for the positions (x, y, θ) . The quotient set $\text{Adj}^+(\sigma)/\sim_\sigma$, containing all the possible surface pairs that we can add to the stance, is in canonical bijection with $p_{\mathbb{N}^4}(E) \setminus p_{\mathbb{N}^4}(\sigma)$, i.e. the set of surface pairs that are not already present in the stance. So for each 4-tuple $(r_1, s_1, r_2, s_2) \in p_{\mathbb{N}^4}(E) \setminus p_{\mathbb{N}^4}(\sigma)$ we denote $\text{cl}_\sigma(r_1, s_1, r_2, s_2)$ the corresponding equivalence class, which contains all the possible positions (x, y, θ) when we want to add the surface pair (r_1, s_1, r_2, s_2) as a contact (this equivalence class is thus homeomorphic to $\mathbb{R}^2 \times \mathbb{S}^1$)

$$\begin{aligned} \text{cl}_\sigma(r_1, s_1, r_2, s_2) = \\ \left\{ \sigma \cup \{(r_1, s_1, r_2, s_2, x, y, \theta)\} \mid (x, y, \theta) \in \mathbb{R}^2 \times \mathbb{S}^1 \right\}. \end{aligned}$$

We now have all the ingredients to write an algorithm that tries to solve Problem 1: exploring $\text{Adj}^-(\sigma)$ is straightforward; for $\text{Adj}^+(\sigma)$, the algorithm explores every equivalence class from $\text{Adj}^+(\sigma)/\sim_\sigma$ by solving an optimization problem on (x, y, θ) .

Before concluding this section, we will state a last useful property related to feasible transitions between two adjacent stances. For two adjacent stances σ and σ' , a configuration in $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ is a configuration that realizes the geometric closure condition for the larger stance of the two ($\mathcal{Q}_{\sigma \cup \sigma'}$) and the feasibility condition for the smaller stance of the two ($\mathcal{F}_{\sigma \cap \sigma'}$). We can formalize this through the following property, illustrated in Fig. 3:

Proposition 1: Let $\sigma \in \Sigma$ and $\sigma' \in \text{Adj}(\sigma)$. Then we have

$$\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'} = \mathcal{Q}_{\sigma \cup \sigma'} \cap \mathcal{F}_{\sigma \cap \sigma'}.$$

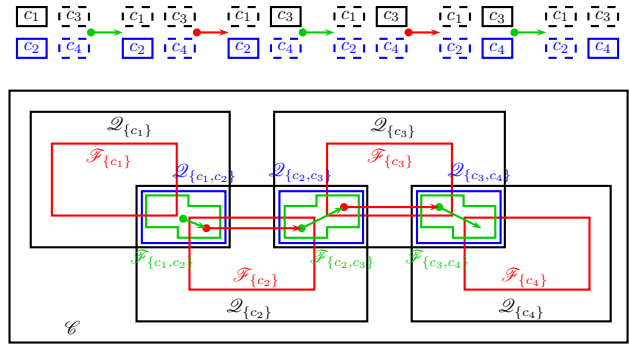


Fig. 4. Transfer and transit paths for a biped feasible sequence of stances. In green transfer paths, in red transit paths. The top figure represents the footprints in Σ (right foot in blue, left foot in black), the bottom figure is a representation in \mathcal{C} (for clarity $\mathcal{Q}_{\{c_1\}} \cap \mathcal{Q}_{\{c_4\}}$ is not represented).

Proof: $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'} \subset \mathcal{Q}_{\sigma \cup \sigma'} \cap \mathcal{F}_{\sigma \cap \sigma'}$ is trivial. Inversely, let $q \in \mathcal{Q}_{\sigma \cup \sigma'} \cap \mathcal{F}_{\sigma \cap \sigma'}$. This implies that q belongs to $\mathcal{Q}_{\sigma \cup \sigma'}$ and is a physically valid static configuration, hence $q \in \mathcal{F}_{\sigma \cup \sigma'}$ and subsequently $q \in \mathcal{F}_{\sigma \cap \sigma'} \cap \mathcal{F}_{\sigma \cup \sigma'} = \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$. ■

Corollary 1: Let $\sigma \in \Sigma$ and $\text{cl}_\sigma(r_1, s_1, r_2, s_2) \in \text{Adj}^+(\sigma)/\sim_\sigma$. Then we have

$$\begin{aligned} \mathcal{F}_\sigma \cap \left(\bigcup_{(x,y,\theta)} \mathcal{F}_{\sigma \cup \{(r_1, s_1, r_2, s_2, x, y, \theta)\}} \right) = \\ \mathcal{F}_\sigma \cap \left(\bigcup_{(x,y,\theta)} \mathcal{Q}_{\sigma \cup \{(r_1, s_1, r_2, s_2, x, y, \theta)\}} \right). \end{aligned}$$

The role of Proposition 1 is to release redundant constraints in Definition 4, while Corollary 1 will prove useful later in the course of this paper (Section III-B).

Remark 4: In some works [16], [10], [17] a path through \mathcal{F}_σ from $q \in \mathcal{F}_\sigma$ to $q' \in \mathcal{F}_\sigma \cap \mathcal{Q}_{\sigma'}$ for $\sigma' \in \text{Adj}^+(\sigma)$ would be called a *transit path*, and a path through \mathcal{F}_σ from $q \in \mathcal{F}_\sigma$ to $q' \in \mathcal{Q}_{\sigma} \cap \mathcal{F}_{\sigma'}$ for $\sigma' \in \text{Adj}^-(\sigma)$ is called a *transfer path* (cf. Fig. 4).

III. ALGORITHM

Our objective now is to solve Problem 1 formulated in Section II.

A. The Discrete Approach

In this section we discuss the approach proposed in the works of Hauser et al. and see how it fits in our generalized formalism for multiple agents. This approach is based on prior discretization of E . Let E_{finite} be a finite subset of E containing the start and goal stances,

$$(\sigma_{\text{start}} \cup \sigma_{\text{goal}}) \subset E_{\text{finite}} \subset E, \text{card}(E_{\text{finite}}) < \infty.$$

Let Σ_{finite} be the restriction of Σ to E_{finite} ,

$$\Sigma_{\text{finite}} = \{\sigma \in \Sigma \mid \sigma \subset E_{\text{finite}}\}.$$

Σ_{finite} is a finite set endowed with a finite undirected graph structure defined by the adjacency relation, as can be seen through the following constructions (“Trans” stands for *transitions* [2])

$$\begin{aligned} \text{Adj}_{\text{finite}}(\sigma) &= \text{Adj}(\sigma) \cap \Sigma_{\text{finite}}, \\ \text{Trans}(\sigma) &= \{\sigma\} \times \text{Adj}_{\text{finite}}(\sigma), \\ \mathcal{G} &= \bigcup_{\sigma \in \Sigma_{\text{finite}}} \text{Trans}(\sigma) \\ &= \{(\sigma_1, \sigma_2) \in \Sigma_{\text{finite}}^2 \mid \sigma_1 \sqsubset \sigma_2 \text{ or } \sigma_2 \sqsubset \sigma_1\}. \end{aligned}$$

These constructions give us the finite graph structure that we wanted $(\Sigma_{\text{finite}}, \mathcal{G})$.

Hauser’s algorithm explores this graph by growing a connected sub-graph $(\mathcal{V}, \mathcal{E})$, $\mathcal{V} \subset \Sigma_{\text{finite}}$, $\mathcal{E} \subset \mathcal{G}$, and maintaining a priority queue $Q \subset \mathcal{G} \times \mathbb{R}$. Let $f : \Sigma_{\text{finite}} \rightarrow \mathbb{R}$ be a cost function on the stances, this cost function is based on different heuristics such as the distance to goal, the distance to reference configurations, and the robustness of the static equilibrium. Algorithm 1 gives the outline of the planner (the *expansion* phase of the multi-modal planner [2]). $p_{\mathcal{G}} : \mathcal{G} \times \mathbb{R} \rightarrow \mathcal{G}$ denotes the canonical projection on \mathcal{G} .

Algorithm 1 FIND_SEQUENCE_OF_STANCES($\sigma_{\text{start}}, \sigma_{\text{goal}}$)

```

1:  $\mathcal{V} \leftarrow \{\sigma_{\text{start}}\}$ 
2:  $\mathcal{E} \leftarrow \emptyset$ 
3:  $Q \leftarrow \emptyset$ 
4: for all  $(\sigma_{\text{start}}, \sigma') \in \text{Trans}(\sigma_{\text{start}})$  do
5:    $Q \leftarrow Q \cup \{(\sigma_{\text{start}}, \sigma', f(\sigma'))\}$ 
6: end for
7: repeat
8:    $(\sigma_{\text{current}}, \sigma_{\text{adjacent}}, c) \leftarrow Q.\text{POP\_LOWEST\_COST}()$ 
9:    $q_{\text{adjacent}} \leftarrow \text{SAMPLE\_RANDOM}(\mathcal{F}_{\sigma_{\text{current}} \cap \sigma_{\text{adjacent}}} \cap \mathcal{Q}_{\sigma_{\text{current}} \cup \sigma_{\text{adjacent}}})$ 
10:  if  $q_{\text{adjacent}} \neq \text{NULL}$  then
11:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{\sigma_{\text{adjacent}}\}$ 
12:     $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\sigma_{\text{current}}, \sigma_{\text{adjacent}})\}$ 
13:    for all  $(\sigma_{\text{adjacent}}, \sigma') \in \text{Trans}(\sigma_{\text{adjacent}}) \setminus p_{\mathcal{G}}(Q)$  do
14:       $Q \leftarrow Q \cup \{(\sigma_{\text{adjacent}}, \sigma', f(\sigma'))\}$ 
15:    end for
16:  else
17:     $Q \leftarrow Q \cup \{(\sigma_{\text{current}}, \sigma_{\text{adjacent}}, c + \text{COST\_INCREMENT})\}$ 
18:  end if
19: until  $\sigma_{\text{goal}} \in \mathcal{V}$  or  $c.\text{IS\_OUT\_OF\_RANGE}()$ 

```

Starting from σ_{start} the algorithm enqueues all the discretized stances that are adjacent to σ_{start} (lines 1 to 5), indifferently adding or removing a contact since they are all in finite number. Then it enters the main search loop (lines 7 to 19): first dequeuing the “most promising” pair of stances made of the currently explored stance along with one of its adjacent stances (line 8), and tries to sample a feasible transition configuration using Proposition 1 (line 9). In case of success (lines 10 to 15), the adjacent stance is added to

the exploration graph (lines 11 and 12) and all the transitions from this adjacent stances (i.e. the stances that are adjacent to the adjacent stance) that are not already present in the queue are enqueued for future exploration (lines 13 to 15). In case of failure to sample a transition configuration, the considered pair is penalised by augmenting its cost and re-enqueued into Q (lines 16 and 17). As the exploration progresses and no solution is found, more time will be allocated to sampling reluctant transitions.

In the worst case, this algorithm will end up exploring all the stances in the connected component of $(\Sigma_{\text{finite}}, \mathcal{G})$ containing σ_{start} . However, if no solution is found then this does not necessarily mean that Problem 1 does not have a solution, but it could also be due to the fact that the discretization performed by choosing E_{finite} might not have been fine enough. This problem is not encountered in our proposed algorithm that we detail hereunder.

B. The Continuous Approach

In this approach we do not need to discretize Σ . We grow a tree $(\mathcal{V}, \mathcal{E})$, $\mathcal{V} \subset \Sigma$, $\mathcal{E} \subset \Sigma^2$, and we maintain on it a priority queue $Q \subset \Sigma \times \mathbb{R}$. Let $f : \mathcal{C} \rightarrow \mathbb{R}$ be a cost function on the configuration space. Algorithm 2 gives the outline of the planner, where ε and δ are two positive parameters, and DISTANCE is a heuristic “metric” on Σ .

Algorithm 2 FIND_SEQUENCE_OF_STANCES($\sigma_{\text{start}}, \sigma_{\text{goal}}$)

```

1:  $q_{\text{start}} \leftarrow \text{FIND\_BEST\_CONFIG}(\mathcal{F}_{\sigma_{\text{start}}})$ 
2:  $\mathcal{V} \leftarrow \{\sigma_{\text{start}}\}$ 
3:  $\mathcal{E} \leftarrow \emptyset$ 
4:  $Q \leftarrow \{(\sigma_{\text{start}}, f(q_{\text{start}}))\}$ 
5: repeat
6:    $(\sigma_{\text{current}}, c) \leftarrow Q.\text{POP\_LOWEST\_COST\_STANCE}()$ 
7:   for all  $\sigma_{\text{adjacent}} \in \text{Adj}^-(\sigma_{\text{current}})$  do
8:      $q_{\text{adjacent}} \leftarrow \text{FIND\_BEST\_CONFIG}(\mathcal{Q}_{\sigma_{\text{current}}} \cap \mathcal{F}_{\sigma_{\text{adjacent}}})$ 
9:     if  $q_{\text{adjacent}} \neq \text{NULL}$  and  $\text{DISTANCE}(\sigma_{\text{adjacent}}, \mathcal{V}) > \varepsilon$  then
10:        $\mathcal{V} \leftarrow \mathcal{V} \cup \{\sigma_{\text{adjacent}}\}$ 
11:        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\sigma_{\text{current}}, \sigma_{\text{adjacent}})\}$ 
12:        $Q \leftarrow Q \cup \{(\sigma_{\text{adjacent}}, f(q_{\text{adjacent}}))\}$ 
13:     end if
14:   end for
15:   for all  $cl_{\sigma_{\text{current}}}(r_1, s_1, r_2, s_2) \in \text{Adj}^+(\sigma_{\text{current}}) / \sim_{\sigma_{\text{current}}}$  do
16:      $q_{\text{adjacent}} \leftarrow \text{FIND\_BEST\_CONFIG}(\mathcal{F}_{\sigma_{\text{current}}} \cap (\bigcup_{(x,y,\theta) \in \mathbb{R}^2 \times \mathbb{S}^1} \mathcal{Q}_{\sigma_{\text{current}} \cup \{(r_1, s_1, r_2, s_2, x, y, \theta)\}}))$ 
17:      $\sigma_{\text{adjacent}} \leftarrow p_{\Sigma}(q_{\text{adjacent}})$ 
18:     if  $q_{\text{adjacent}} \neq \text{NULL}$  and  $\text{DISTANCE}(\sigma_{\text{adjacent}}, \mathcal{V}) > \varepsilon$  then
19:        $\mathcal{V} \leftarrow \mathcal{V} \cup \{\sigma_{\text{adjacent}}\}$ 
20:        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\sigma_{\text{current}}, \sigma_{\text{adjacent}})\}$ 
21:        $Q \leftarrow Q \cup \{(\sigma_{\text{adjacent}}, f(q_{\text{adjacent}}))\}$ 
22:     end if
23:   end for
24: until  $\text{DISTANCE}(\sigma_{\text{goal}}, \mathcal{V}) < \delta$  or  $Q = \emptyset$ 

```

First, the algorithm enqueues σ_{start} (lines 1 to 4). Then it enters the main search loop (lines 5 to 24), which consists once again in dequeuing the “most promising” stance (line 6), and exploring the adjacent stances. This exploration is split into two stages: the adjacent stances by removing a contact (lines 7 to 14) and the adjacent stances by adding a contact (lines 15 to 23). The former adjacent stances are in finite number and for each of them the algorithm tries to sample a feasible transition configuration (line 8). In case of success, the adjacent stance, if not already in the exploration graph, is added to this exploration graph and enqueued (lines 9 to 13). The latter adjacent stances are explored via their equivalence classes, meaning that the algorithm picks up a pair of surfaces not already in the currently explored stance (line 15), and for every such pair it tries to find a transition configuration while simultaneously looking for the best relative position for the pair of surfaces (line 16), upon success the pair of surfaces is completed as a contact with the found relative position and added to the current stance (line 17) to form the adjacent stance that will be enqueued and added the exploration graph if not already present (lines 18 to 22).

The main added value of Algorithm 2 with regard to Algorithm 1 lies in line 16. Indeed, both Algs. 1 and 2 rely on an *inverse stance solver* that returns configurations from 3 types of queries:

- type 1 queries are made on spaces of the form \mathcal{F}_σ ,
- type 2 queries are made on spaces of the form $\mathcal{Q}_\sigma \cap \mathcal{F}_{\sigma'}$, where $\sigma' \in \text{Adj}^-(\sigma)$ (cf. Proposition 1),
- type 3 queries are made on spaces of the form $\mathcal{F}_\sigma \cap \left(\bigcup_{(x,y,\theta)} \mathcal{Q}_{\sigma \cup \{(r_1, s_1, r_2, s_2, x, y, \theta)\}} \right)$ where $\text{cl}_\sigma(r_1, s_1, r_2, s_2) \in \text{Adj}^+(\sigma) / \sim_\sigma$ (cf. Corollary 1).

In Algorithm 1 this inverse stance solver is called through `SAMPLE_RANDOM` and is based the *Iterative Constraint Enforcement* method described in [9]. In Algorithm 2 the solver is called through `FIND_BEST_CONFIG`. It is based on a “black-box” non-linear optimization solver, detailed in [18]. While type 2 queries are answered by both solvers, processing type 3 queries is a specificity of our solver, which, for $\sigma \in \Sigma$ and $\text{cl}_\sigma(r_1, s_1, r_2, s_2) \in \text{Adj}^+(\sigma) / \sim_\sigma$, solves the following optimization problem

$$\begin{aligned} & \min_{q, (x, y, \theta)} \text{obj}(q) \\ & \text{subject to } \begin{cases} (x, y, \theta) = p_{\mathbb{R}^2 \times \mathbb{S}^1}(q) \\ q \in \mathcal{F}_\sigma \\ q \in \mathcal{Q}_{\sigma \cup \{(r_1, s_1, r_2, s_2, x, y, \theta)\}}, \end{cases} \end{aligned}$$

where $p_{\mathbb{R}^2 \times \mathbb{S}^1} : \mathcal{C} \rightarrow \mathbb{R}^2 \times \mathbb{S}^1$ is the “forward kinematics” mapping which inverts for (x, y, θ) the contact equations (2), (3), and (4). The objective function to minimize $\text{obj}(q)$ takes the form

$$\begin{aligned} \text{obj}(q) = & (q - q_{\text{goal}})^T A (q - q_{\text{goal}}) \\ & + (o_{r_1, s_1}(q) - o_{r_1, s_1}(q_{\text{goal}}))^T B_1 (o_{r_1, s_1}(q) - o_{r_1, s_1}(q_{\text{goal}})) \\ & + (o_{r_2, s_2}(q) - o_{r_2, s_2}(q_{\text{goal}}))^T B_2 (o_{r_2, s_2}(q) - o_{r_2, s_2}(q_{\text{goal}})), \end{aligned}$$

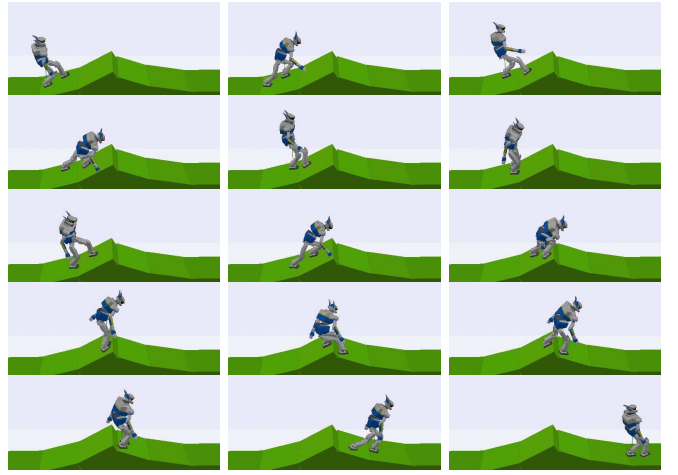


Fig. 5. Biped locomotion over irregular terrain. Coulomb friction allows the robot not to slip. The friction coefficient is set to $\mu = 1$.

where A, B_1, B_2 are symmetric positive semi-definite matrices and q_{goal} is either a configuration from $\mathcal{F}_{\sigma_{\text{goal}}}$ or an intermediate milestone from a *guide path* given by a collision-free path planner detailed in our previously published work [19].

Algorithm 2 is a best-first search algorithm. As such, it is a greedy algorithm that suffers from the local minima problem. To avoid this, many heuristics can be added to the algorithmic blueprint defined by Algorithm 2 [1], [4], [5], [19]. However, anecdotally, such problems were not encountered in the runs of the planner that we made in the experiments of Section IV. Although completeness and global optimality issues are not tackled in our work, the analysis here being only qualitative, the proposed algorithm proved to be practically efficient in solving the queries of Section IV.

IV. RESULTS

In this section we show results obtained by applying the generic algorithm Algorithm 2 to different classes of problems, cf. Figs. 5, 6, 7, 8, and 9. In all these figures, for the computed solution sequence of stances $(\sigma_1, \dots, \sigma_n) \in \Sigma^n$ of the considered problem, we display a sample subsequence of a sequence of configurations $(q'_1, \dots, q'_n) \in \mathcal{C}^n$ such that $q'_1 \in \mathcal{F}_{\sigma_1}$ and $\forall i \in \{2, \dots, n\} q'_i \in \mathcal{F}_{\sigma_i} \cap \mathcal{F}_{\sigma_{i-1}}$. It is very important to emphasize here that the pictures are not snapshots of a continuous motion. They are not merely representative of the result, they are the result. So it is important to keep this in mind in order not to over-estimate the results presented here.

In these scenarios we used three robots models:

- a model of the HRP-2 humanoid robot [20] appearing in Figs. 5, 7, 8, and 9,
- rigid objects: the ball of Fig. 6, the table of Fig. 7, and the box of Fig. 8,
- fixed-base manipulators: the four fingers of Fig. 6.

Surface patches on the robots have been chosen as follows:

- one surface per foot of the HRP-2 robot in all the scenarios, one surface per hand in Figs. 7, 8, and 9,

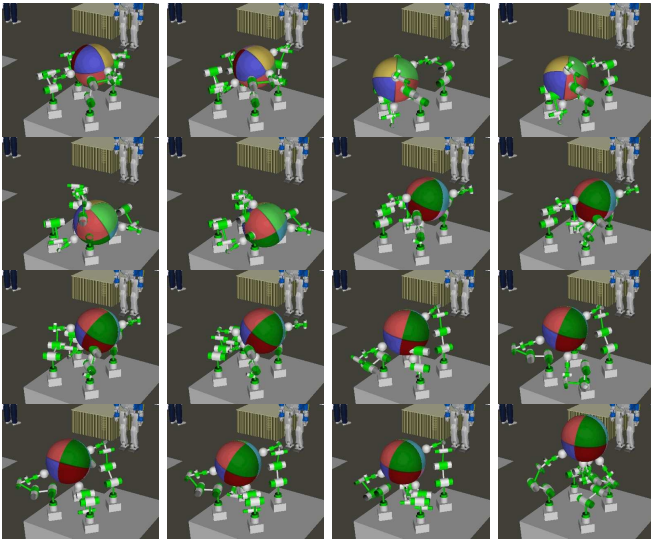


Fig. 6. Dexterous manipulation. The objective is to rotate the 3 kg ball upside down. The fingers are 6-DOF elbow-like manipulators with wrist-like end-effectors. The friction coefficients between the end-effectors of the fingers and the ball are set to $\mu = 1$. No limits are considered on the torques delivered by the actuators in the fingers.

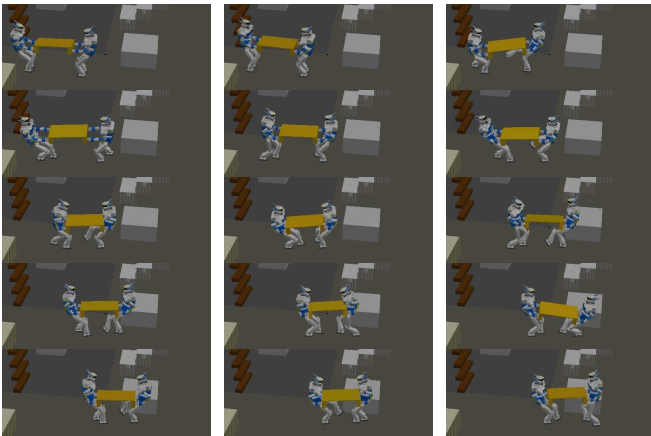


Fig. 7. Collaborative manipulation. Here we use an improved version of Algorithm 2 as contacts between the hands of the robots and the handles of the table are required not to be broken during the planning, as specified at problem-instantiation-time by the user.

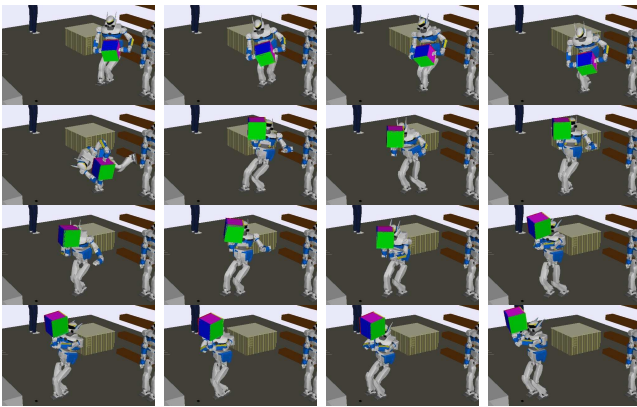


Fig. 8. Combined whole-body manipulation and locomotion. The objective is for the HRP-2 robot to advance 2 m forward while simultaneously performing half rotation of the 5 kg box, bringing the purple face up. Friction coefficients between the hands and the box are set to $\mu = 1$.

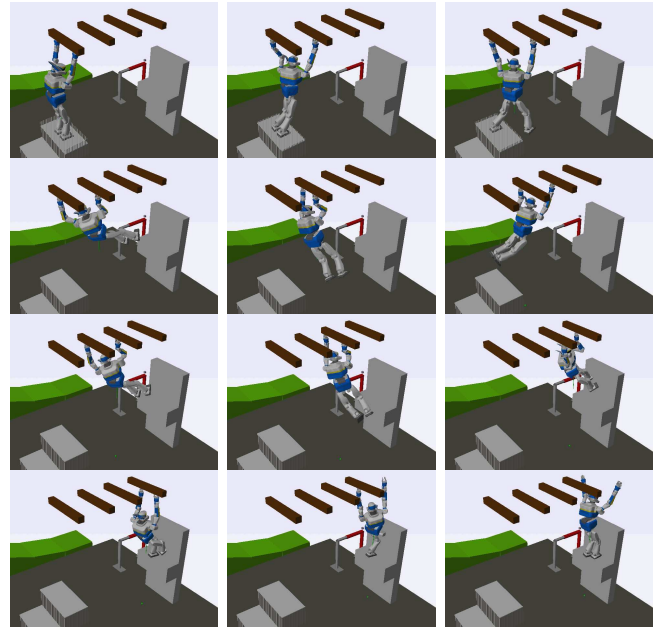


Fig. 9. Bilateral contacts on monkey bars. This example illustrates the necessity of use of bilateral contacts to solve the planning problem.

- one surface per planar piece of the ground in all the scenarios,
- one surface per face of the cube in Fig. 8,
- one surface per handle of the table in Fig. 7,
- one surface per monkey bar in Fig. 9,
- one surface per fingertip in Fig. 6,
- 20 regularly distributed planar surfaces tangent to the ball in Fig. 6. Every such plane approximates the spherical surface around the tangent point. Contacts yielded on this tangent planes are then projected back onto the spherical surface.

In the modeling of the feasible spaces \mathcal{F}_σ we considered the following constraints [18]:

- static equilibrium for all the underactuated free-base robots (including objects) considered as individual systems, under the action of external contact forces, gravity force, and actuation torques,
- Newton's third law for all the internal contact forces on the system of robots and objects considered as a whole,
- Coulomb friction model for the unilateral contact forces (all the forces except the ones listed in the next item),
- fixed grasp model for the bilateral contact forces: the contacts between the hands of the robots and the handles of the table in Fig. 7, and between the hands of the robot and the monkey bars in Fig. 9,
- joint angles limits for all the joints of the poly-articulated mechanisms (HRP-2 and the multi-fingered hand),
- bounds on the torques of all the actuators in HRP-2, except for the wrist actuators.

However, collision avoidance constraints have not yet been taken into account in our current implementation of the feasible spaces. This did not affect the scenarios that we

TABLE I
EXPERIMENTAL RESULTS

	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 9
N (robots)	1	5	3	2	1
$\dim(\mathcal{C})$	46	30	98	52	46
Num. of steps	32	17	26	24	33
Size of the tree	51	846	47	144	91
Comp. time (s)	133	830	318	230	750

have chosen on purpose, but we are currently working on integrating these constraints in our inverse stance solver. Note also that, when applicable, the scenarios were chosen to demonstrate the performance of the planner in situations in which friction is specifically required to solve the problem, as highlighted by a relatively high coefficient of friction ($\mu = 1$). Such a high friction coefficient is required for example to cross the steepest part of the hill in Fig. 5 (as opposed to standing on horizontal planar surface in which low friction is enough), or to manipulate the box using only planar unilateral contact in Fig. 8 without resorting to bilateral grasps. Lower coefficient of friction would be sufficient for less constraining problems.

Tab. I gives some experimental figures concerning these scenarios made on a 3.06 GHz computer running under Windows XP. The program is compiled from a C++ implementation of the framework.

V. CONCLUSION

We wrote a multi-contact stances planning algorithm for multiple robots having to make use of contacts to perform locomotion or manipulation tasks. The autonomy of the robots is enhanced as little domain knowledge is required to plan an acyclic non-gaited sequence of stances. This autonomy is further increased by not specifying pre-discretized candidate contact locations on the environment, the continuity of which is totally explored by the planner. Along with autonomy, the other key driving concept of this work was the generality. Our planner was not targeted for any specific model of robot or system of robots. The planner successfully performed on a set of problems taken from different sub-fields of motion planning in robotics, namely, the legged locomotion, dexterous manipulation, combined whole-body locomotion and manipulation, and collaborative manipulation problems. All these locomotion and manipulation problems were unified within the same framework.

The next step is to take the output of this algorithm as an input of a motion planning algorithm that would plan the continuous motion going through these stances. Although static criteria were considered in the stances planning stage, the continuous motion planner can use them, along with the generated configurations that correspond to each stance of the plan, as milestones to plan a dynamic trajectory. At this latter stage, the kinematics of changing contact modes such as sliding, pure rolling, etc. as listed in [12] can also be considered, they could not be taken into account within our static formulation that was designed for planning

static stances with no kinematics or dynamics considerations. These are our current subjects of research.

ACKNOWLEDGEMENT

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

REFERENCES

- [1] A. Escande, "Contact planning for acyclic motion with application to humanoids," Ph.D. dissertation, University of Evry-Val d'Essone, December 2008.
- [2] K. Hauser, "Motion planning for legged and humanoid robots," Ph.D. dissertation, Stanford University, December 2008.
- [3] K. Hauser and J.-C. Latombe, "Multi-modal motion planning for non-expansive spaces," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2008.
- [4] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on HRP-2," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.
- [5] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning support contact-points for acyclic motions and experiments on HRP-2," in *Proceedings of the International Symposium on Experimental Robotics*, 2008.
- [6] A. Escande and A. Kheddar, "Contact planning for acyclic motion with tasks constraints," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.
- [7] J. C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [8] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [9] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2005.
- [10] J.-P. Saut, A. Sahbani, S. El-Khoury, and V. Perdereau, "Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.
- [11] J. Xu, J. Koo, and Z. Li, "Finger gaits planning for multifingered manipulation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.
- [12] M. Yashima, Y. Shiina, and H. Yamaguchi, "Randomized manipulation planning for a multifingered hand by switching contact modes," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.
- [13] A. Miller and P. K. Allen, "Grasplit!: A versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 110–122, December 2004.
- [14] C. Esteves, G. Arechavelata, J. Pettré, and J.-P. Laumond, "Animation planning for virtual characters cooperation," *ACM Transactions on Graphics*, vol. 25, no. 2, pp. 319–339, April 2006.
- [15] D. J. Montana, "The kinematics of contact and grasp," *International Journal of Robotics Research*, vol. 7, no. 3, pp. 17–32, June 1988.
- [16] R. Alami, J.-P. Laumond, and T. Siméon, "Two manipulation planning algorithms," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 1995.
- [17] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, July-August 2004.
- [18] K. Bouyarmane and A. Kheddar, "Static multi-contact inverse problem for multiple humanoid robots and manipulated objects," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2010.
- [19] K. Bouyarmane, A. Escande, F. Lamiraux, and A. Kheddar, "Potential field guide for multicontact humanoid motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.
- [20] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.