Potential Field Guide for Humanoid Multicontacts Acyclic Motion Planning

Karim Bouyarmane
CNRS-UM2 LIRMM, France
CNRS-AIST JRL, Japan

Adrien Escande
CEA LIST, France
CNRS-AIST JRL, Japan

Florent Lamiraux
CNRS LAAS, France
CNRS-AIST JRL, Japan

Abderrahmane Kheddar
CNRS-UM2 LIRMM, France
CNRS-AIST JRL, Japan

***Abstract*—We present a motion planning algorithm that computes rough trajectories used by a contact-points planner as a guide to grow its search graph. We adapt collision-free motion planning algorithms to plan a path within the guide space, a submanifold of the configuration space included in the free space in which the configurations are subject to static stability constraint. We first discuss the definition of the guide space. Then we detail the different techniques and ideas involved: relevant C-space sampling for humanoid robot, task-driven projection process, static stability test based on polyhedral convex cones theory's double description method. We finally present results from our implementation of the algorithm.**

I. Introduction

Contact-points planning is a motion planning approach that aims at overcoming difficulties of cyclic gaited humanoid motion planning in unstructured and highly constrained environments. Examples of such planners are presented in [1] [2]. In [2] Best First Planning was adapted by growing the search tree in the space of sets of contacts. A key element of this contacts planner is the potential field that drives the search. It has to be carefully chosen as the planner may get trapped in local minima, which occur for example when we choose too simple potential fields such as the Euclidian distance to goal. An inappropriate potential field may also lead to the planning of complicated paths and postures. In [3], a solution is given by building the potential field around a rough trajectory, a *contact-points guide*, that gives an approximation of the intended path in the workspace as well as an idea of the postures that the robot has to adopt along this path. This trajectory was given manually as an input to the planner. Our aim in this work is to provide such a trajectory automatically, thus giving more autonomy to the robot.

II. Solution

The main idea is to adapt existing collision-free motion planning algorithms to plan the contact-points guide.

A. General algorithm

The collision-free motion planning problem can be formalized as follows (adapted from [4]):

Formulation 1 (collision-free motion planning problem)**.**

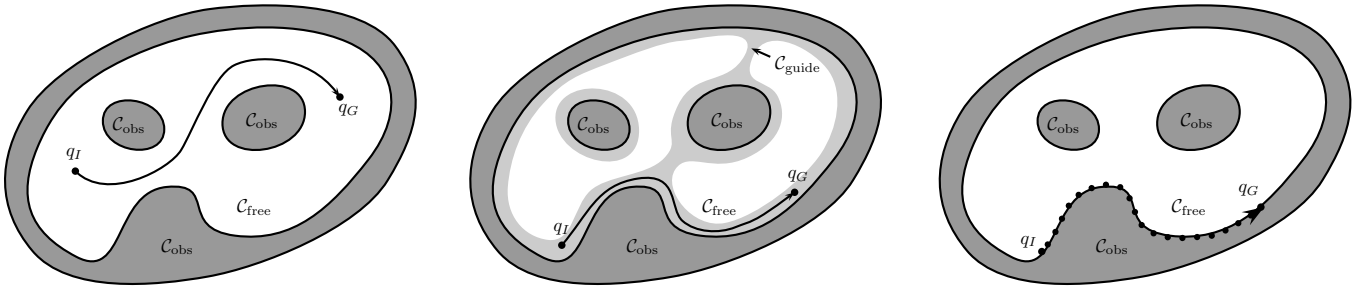
- a *world* $\mathcal{W} = \mathbb{R}^3$.
- an *obstacle region* $\mathcal{O} \subset \mathcal{W}$.
- a *robot* $\mathcal{R}$ defined in $\mathcal{W}$ as a kinematic tree of m *joints* $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_m$ to which *rigid bodies* $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$ are attached.
- the *configuration space* (also called *C-space*) $\mathcal{C}$ defined as the set of all possible transformations that may be applied to the robot. The image of the robot $\mathcal{R}$ in the configuration q is denoted $\mathcal{R}(q)$. From $\mathcal{C}$ we derive $\mathcal{C}_{\text{free}} = \{q \in \mathcal{C} \mid \mathcal{R}(q) \cap \mathcal{O} = \varnothing\}$ and $\mathcal{C}_{\text{obs}} = \mathcal{C} \setminus \mathcal{C}_{\text{free}}$.
- a *query pair* $(q_I, q_G) \in \mathcal{C}_{\text{free}}^2$ of initial and goal configurations.
- an algorithm must compute a continuous *path* $\tau : [0,1] \to \mathcal{C}_{\text{free}}$ such that $\tau(0) = q_I$ and $\tau(1) = q_G$.

Two classes of methods exist so far to address this problem [4]: combinatorial motion planning and sampling-based motion planning. The difference between the two lies in that the latter avoids explicit construction of $\mathcal{C}_{\text{obs}}$. Instead it uses a sampling of the C-space to grow a discrete graph $\mathcal{G}(V, E)$, called a *roadmap*, of which every vertex $v \in V$ represents a configuration $q \in \mathcal{C}_{\text{free}}$ and every edge $e \in E$ represents a continous path in $\mathcal{C}_{\text{free}}$, that progressively covers $\mathcal{C}_{\text{free}}$. The search for the path is then conducted into the constructed roadmap that supposedly represents an approximation of the connectivity of $\mathcal{C}_{\text{free}}$. Different instantiations of sampling-based motion planning as a general approach exist [5] [6]. Algorithm 1 gives the general frame of the one we take as a starting point for our study, keeping in mind that it is possible to choose any other instantiation modulo adequate modifications.

Algorithm 1 sampling-based collision-free motion planning.

```
initialize G(V ← {q_I, q_G}, E ← ∅)
while no path found in G do
    sample a random configuration q_s in C
    if q_s ∈ C_free then
        for all q_V ∈ V ∩ NEIGHBOURHOOD(q_s) do
            if the direct path τ_d(q_s, q_V) lies in C_free then
                V.add(q_s) and E.add(τ_d(q_s, q_V))
            end if
        end for
    end if
end while
```

Now we would like to adapt algorithm 1 in order to plan a



(a) Collision-free motion planning. (b) Contacts guide planning. (c) Contact-points planning.

Fig. 1. Illustration of the problem.

contact-points guide. The problem is that the path yielded by a contact-points planner lies on the *boundary* of $\mathcal{C}_{\text{obs}}$: $\partial\mathcal{C}_{\text{obs}}$. Simply replacing $\mathcal{C}_{\text{free}}$ with $\partial\mathcal{C}_{\text{obs}}$ in algorithm 1 would be a failing strategy as the measure of $\partial\mathcal{C}_{\text{obs}}$ is equal to zero . This means that the rejection rate at line 4 would be equal to 1. The second problem with this strategy concerns the linear direct paths in line 6, as $\partial\mathcal{C}_{\text{obs}}$ is generally a non linear submanifold, a linear edge joining two of its elements will almost always be completely outside the submanifold.

Our solution is to consider a submanifold of $\mathcal{C}$ of non-zero measure, we label it $\mathcal{C}_{\text{guide}}$, that can be visually represented as a layer wrapping each connected component of $\partial\mathcal{C}_{\text{obs}}$. The idea, to some extent similar to [7], is to sample configurations "near" the obstacles; however, work in [7] focuses on 6D rigid robots, whereas our primary targets are polyarticulated humanoid robots. We will now detail our definition of $\mathcal{C}_{\text{guide}}$.

A *contact situation* between a body $\mathcal{B}_i$ of the robot and the obstacle region $\mathcal{O}$ is normally defined as

$$\partial\mathcal{B}_i \cap \partial\mathcal{O} \neq \varnothing \quad \text{and} \quad \text{int}(\mathcal{B}_i) \cap \text{int}(\mathcal{O}) = \varnothing$$

One way of adding a dimension, and thus creating a "volume", to the submanifold $\mathcal{C}_{\text{guide}}$ could be to consider the body $\mathcal{B}_i$ as in contact with $\mathcal{O}$ if $d(\mathcal{B}_i, \mathcal{O}) < \varepsilon_{\text{contact}}$, which is a positive fixed threshold. d denotes the Euclidian distance.

Definition 1 (body-obstacle contact situation)**.** A rigid body $\mathcal{B}$ is in contact with an obstacle region $\mathcal{O}$ if

$$0 < d(\mathcal{B}, \mathcal{O}) < \varepsilon_{\text{contact}}$$

In this situation, we denote by $A_{\mathcal{B}}$ and $A_{\mathcal{O}}$ respectively the closest points on the body and on the obstacle and by $\mathbf{n} = \overrightarrow{A_{\mathcal{O}}A_{\mathcal{B}}}/\|\overrightarrow{A_{\mathcal{O}}A_{\mathcal{B}}}\|$ the normal of the contact. The robot $\mathcal{R}$ is in contact in configuration $q \in \mathcal{C}_{\text{free}}$ if at least one of its bodies is in contact in configuration q.

We can now define $\mathcal{C}_{\text{guide}}$ as

$$\mathcal{C}_{\text{guide}} = \{q \in \mathcal{C}_{\text{free}} \mid \mathcal{R} \text{ is in contact in configuration } q\}$$

and then plan a collision-free path in $\mathcal{C}_{\text{guide}}$ using algorithm 1 and replacing in it all the occurrences of $\mathcal{C}_{\text{free}}$ by $\mathcal{C}_{\text{guide}}$. This would produce a path that could be tricky to follow by the contact-points planner [3] as the latter will have to compute *statically stable* configurations along this path, and may need to stray significantly from the given path to find these stable configurations. So we have to refine the definition of $\mathcal{C}_{\text{guide}}$ to take static stability into account.

Considering the laws of rigid body dynamics applied to $\mathcal{R}$ and assuming that there are no limits to the torques we can apply to the robot joints (which is only an approximation), the static stability condition is simply written

$$\begin{cases} \sum_{\mathbf{f}\in\mathcal{F}} \mathbf{f} + m\mathbf{g} = \mathbf{0} \\ \sum_{\mathbf{f}\in\mathcal{F}} \mathscr{M}_O(\mathbf{f}) + \mathscr{M}_O(m\mathbf{g}) = \mathbf{0} \end{cases}$$

where $\mathcal{F}$ is the set of all *contact forces* applied to the robot, and $\mathscr{M}_O$ is the moment of a force in a point $O \in \mathbb{R}^3$. m is the mass of the robot and $\mathbf{g}$ the gravity vector. For simplicity we have modeled any surface contact as a discrete set of punctual contacts applied at chosen points distributed over the contact surface (we intentionally do not make it explicit in our formulas for readability's sake). Each contact force $\mathbf{f} \in \mathcal{F}$ applied on the robot at a point $A \in \partial\mathcal{R}$ with a normal $\mathbf{n}$ lies in a *friction cone* $\mathscr{C}_{A,\mathbf{n},\theta}$, θ being the angle of the cone that depends on the friction coefficient between the body and the obstacle, A is the apex of the cone, and $\mathbf{n}$ defines the revolution axis of the cone.

Definition 2 (static stability situation)**.** The robot $\mathcal{R}$ placed in a configuration $q \in \mathcal{C}_{\text{free}}$ is statically stable if

$$\forall i \in I(q),\ \exists\, \mathbf{f}_i \in \mathscr{C}_{A_{\mathcal{B}_i},\mathbf{n}_i,\theta_i},$$
$$\text{s.t.} \begin{cases} \sum_{i\in I(q)} \mathbf{f}_i + m\mathbf{g} = \mathbf{0} \\ \sum_{i\in I(q)} \mathscr{M}_O(\mathbf{f}_i) + \mathscr{M}_O(m\mathbf{g}) = \mathbf{0} \end{cases}$$

where

$$I(q) = \left\{ i \in \{1, \ldots, m\} \mid 0 < d(\mathcal{B}_i(q), \mathcal{O}) < \varepsilon_{\text{contact}} \right\}$$

We can now introduce our new definition of $\mathcal{C}_{\text{guide}}$ as

$$\mathcal{C}_{\text{guide}} = \{q \in \mathcal{C}_{\text{free}} \,|\, \mathcal{R} \text{ is statically stable in configuration } q\}$$

and once again try to adapt algorithm 1. This is still not enough, as the rejection rate of our sampling would still be very high. This is the reason why we have decided to split the sampling procedure into two distinct phases: the sampling of a more or less uniform random configuration q_{s} in $\mathcal{C}$, followed

by a projection process of q_s to try to make it fit inside $\mathcal{C}_{\text{guide}}$. This projection process is for now only applied on the sampled configurations, and on some discretization points along the linear direct path. There is no guaranty, however, that the whole continuous direct path is inside $\mathcal{C}_{\text{guide}}$.

Finally, we get algorithm 2, which is the adaptation of algorithm 1 taking into account the previously discussed points. $p : \mathcal{C}_{\text{free}} \longrightarrow \mathcal{C}_{\text{guide}}$ denotes the projection function.

Algorithm 2 contact-points guide planning

```
1: initialize G(V ← {q_I, q_G}, E ← ∅)
2: while no path found in G do
3:    sample a random configuration q_s in C
4:    if q_s ∈ C_free then
5:       apply projection q_p = p(q_s) ∈ C_guide
6:       for all q_V ∈ V ∩ NEIGHBOURHOOD(q_p) do
7:          if (a discretization of) τ_d(q_p, q_V) lies in C_guide
            then
8:             V.add(q_p) and E.add(τ_d(q_p, q_V))
9:          end if
10:      end for
11:   end if
12: end while
```

We will now get into the detail of the different steps of execution of algorithm 2, especially the lines 3 and 5.

B. Sampling random configurations

In this section we detail line 3 of algorithm 2.

Our humanoid robot $\mathcal{R}$ is represented as a kinematic tree of m joints $\mathcal{J}_1, \ldots, \mathcal{J}_m$. The root joint $\mathcal{J}_1$ is a six-dimensional free flyer that evolves in the C-space $\mathbb{R}^3 \times SO(3)$, or, if the translations are bounded, $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}] \times SO(3)$. The remaining joints are revolute joints yielding the C-space $\prod_{i=2}^{m} [\theta_{i,\min}, \theta_{i,\max}]$. The total C-space is consequently

$$\mathcal{C} = \mathbb{R}^3 \times SO(3) \times \prod_{i=2}^{m} [\theta_{i,\min}, \theta_{i,\max}]$$

that we can write in a more expressive way as

$$\mathcal{C} = \mathcal{C}_{\text{position}} \times \mathcal{C}_{\text{orientation}} \times \mathcal{C}_{\text{posture}}$$

A random C-space variable Q is as such a vector of three independent random C-space variables $Q = (Q_{\text{position}}, Q_{\text{orientation}}, Q_{\text{posture}})$.

1) Position sampling: Q_{position} can be either a uniform random variable if the workspace $\mathcal{W}$ is bounded or a spatial Gaussian random variable otherwise.

2) Orientation sampling: For the orientation we would like to bias the sampling in order to favor some interesting orientations for a humanoid robot, such as the standing-up orientation for a walk, the laying-down orientation for a crawl, or a slightly front-leant orientation for a climb. $SO(3)$ being homeomorphic to the unit quaternion sphere $\mathbb{S}^3$, we need a random variable that looks like a Gaussian distribution on the sphere $\mathbb{S}^3$ around one of its points q_0 that would represent one of the orientations above. The *Von Mises - Fisher distribution* [8] achieves this very purpose. Given a *mean* unit vector q_0 and a *concentration parameter* $\kappa \in \mathbb{R}^+$, the probability density function of the Von Mises-Fisher distribution on the sphere $\mathbb{S}^{p-1} \subset \mathbb{R}^p$ is

$$f_{q_0,\kappa}(q) = C_p(\kappa) \exp\left(\kappa q_0^T q\right)$$

$C_p(\kappa)$ is a normalization constant

$$C_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)}$$

where I_v denotes the modified Bessel function of the first kind and order v. The parameter κ controls the concentration of the distribution around q_0. The bigger κ the more concentrated the distribution. $\kappa = 0$ yields a uniform distribution over the sphere. An algorithm for simulating a Von Mises-Fisher random variable is given in [9].

3) Posture sampling: Now we want to sample the posture space $\mathcal{C}_{\text{posture}} = \prod_{i=2}^{m} [\theta_{i,\min}, \theta_{i,\max}]$. We could immediately choose for Q_{posture} a uniform random variable. However, this would produce postures that once again are not interesting enough for a humanoid robot, especially when the dimension $m - 1$ of this manifold is relatively high ($m - 1 = 30$ in our humanoid platform). To solve this problem we choose to reduce the dimensionality of $\mathcal{C}_{\text{posture}}$ by sampling in the affine space generated by the standing-up posture q_{key_0} and a certain number of *key postures* $q_{\text{key}_1}, \ldots, q_{\text{key}_n}$. These latter postures should be relevant for a humanoid robot and could represent for example the sitting-down posture, the four-legged posture, etc. To remain within the joints limits, we consider the bounded space

$$\mathcal{C}_{\text{posture}} = \left\{ q_{\text{key}_0} + \sum_{i=1}^{n} \lambda_i (q_{\text{key}_i} - q_{\text{key}_0}) \mid (\lambda_i)_i \in (\mathbb{B}_k^n)^+ \right\}$$

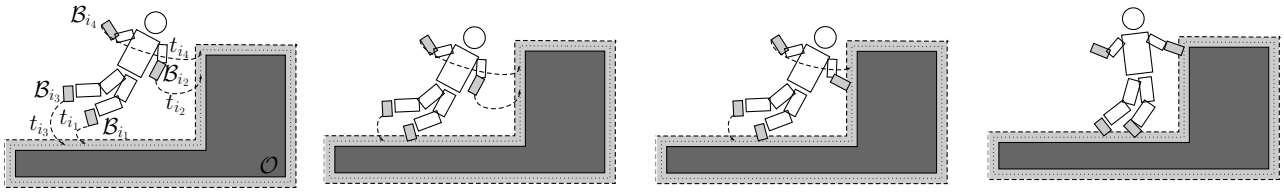
where $(\mathbb{B}_k^n)^+$ is the positive quadrant of the unit ball of dimension n for the k-norm $\|.\|_k$

$$(\mathbb{B}_k^n)^+ = \left\{ (\lambda_i)_i \in [0,1]^n \mid \sum_{i=1}^{n} \lambda_i^k \leq 1 \right\}$$

that we sample uniformly.

C. Projection process

We detail now line 5 of algorithm 2. What we mean by *projection* here is an operation that tries to bring a given configuration sample in $\mathcal{C}_{\text{free}}$ inside $\mathcal{C}_{\text{guide}}$. The idea of projection was introduced in [10] and further investigated in [11]. The solution we choose is to use a *stack of tasks* solver based on *generalized inverse kinematics* called *hppGik* and presented in [12]. A *task* is a function $f : \mathcal{C} \longrightarrow \mathbb{R}$ that we would like to bring to zero, *i.e* to solve $f(q) = 0, q \in \mathcal{C}$. Suppose we have sampled a random configuration q_s. From this configuration we want to compute a statically stable configuration, thus we have to create contacts with the neighboring obstacles, given that the more contacts we create the more stable the configuration



(a) Initial configuration. (b) First iteration. (c) Second iteration. (d) Final configuration statically stable.

Fig. 2. Illustration of the projection process.

is likely to be. On the other hand, the more contacts we create the more we deform the original posture and reduce the mobility for the next posture, this is why we should create the "minimum" number of contacts to ensure the stability. To create a contact between a body $\mathcal{B}$ and the obstacle region $\mathcal{O}$ we need to bring it to a distance closer than $\varepsilon_{\text{contact}}$. Let us define the *goal point* A_{goal} as the point translated from $A_{\mathcal{O}}$ by a $\varepsilon_{\text{contact}}/2$ distance following $\mathbf{n}$, and the *goal plan* P_{goal} as the plan normal to $\mathbf{n}$ in A_{goal}. The task that we want to formalise is "bring the point $A_{\mathcal{B}}$ in the plan P_{goal}", *i.e.* bring to 0 the corresponding task function

$$f(q) = (\overrightarrow{A_{\text{goal}}A_{\mathcal{B}}(q)}|\mathbf{n})$$

where $(.|.)$ denotes the Euclidian scalar product. To solve the task $f(q) = 0$ we implent the *Newton's method* for finding zeros of a function (the same idea is suggested [1]). To do so we linearize f around a start configuration q_0 as

$$f(q) \simeq f(q_0) + \frac{\partial f}{\partial q}(q_0).dq$$

where $dq = q - q_0$ and then we solve the linear system

$$f(q_0) + \frac{\partial f}{\partial q}(q_0).dq = 0$$

using generalized inverse kinematics to compute the pseudo-inverse of $J(q_0) = \frac{\partial f}{\partial q}(q_0)$ that we denote $J(q_0)^\dagger$.The solution q_1 of the system is thus given by

$$q_1 = q_0 - J(q_0)^\dagger f(q_0)$$

The Newton's method consists in iterating again starting now from q_1, meaning that we construct a sequence $(q_n)_{n\in\mathbb{N}}$ recursively as

$$q_{n+1} = q_n - J(q_n)^\dagger f(q_n)$$

that supposedly converges to the solution. However, in our task of bringing the body close to the obstacle, we do not really need to converge to the exact solution, but rather to converge towards a static stability situation, even though this latter is far from the exact solution. This is why we have chosen the Newton's method, as we can stop its execution after each single iteration to test the static stability of the intermediate solutions, and can reach the static stability after few iterations. Now we would like to bring not only one body $\mathcal{B}$ close to the obstacle region $\mathcal{O}$, but the maximum number of bodies $\mathcal{B}_1, \ldots, \mathcal{B}_m$ to $\mathcal{O}$, this means that we need to solve the system of equations:

$$\bigcap_{i=1}^{m} f_i(q) = 0$$

or the linearized version

$$\bigcap_{i=1}^{m} f_i(q_0) + \frac{\partial f_i}{\partial q}(q_0)dq = 0$$

The stack of tasks solver *hppGik* [12] allows us to solve such a system with *priorities*, meaning that it solves the first equation, then it tries to solve the second equation at best while remaining in the solution space of the first equation, and so on. The priority we choose is the distance to obstacle, as we try to bring closer with the highest priority the closest body to the obstacles. Let $i_1, \ldots, i_m \in \{1, \ldots, m\}$ be the indexes of the bodies sorted in increasing order of distance to $\mathcal{O}$, *i.e*:

$$d(\mathcal{B}_{i_1}, \mathcal{O}) \leq d(\mathcal{B}_{i_2}, \mathcal{O}) \leq \ldots \leq d(\mathcal{B}_{i_m}, \mathcal{O})$$

The *hppGik* solver solves, in the order of priority, the following stack of tasks:

$$\bigcap_{j=1}^{m} t_j : f_{i_j}(q_0) + \frac{\partial f_{i_j}}{\partial q}(q_0)dq = 0$$

where t_j is the task of priority j.

Finally we give algorithm 3 of the projection process, in which we introduce one new task per iteration in order to deform as little as possible the posture. We also stop the process after a maximum number of iterations, after which we discard the current configuration and we start again the process with a new q_{s} according to algorithm 2.

Algorithm 3 projection process

1: sample a random configuration q_{s}
2: set $q_0 \leftarrow q_{\text{s}}$
3: COUNTER $\leftarrow 1$
4: **while** q_0 is not statically stable and COUNTER $<$ MAX_ITERATIONS **do**
5: sort the bodies $d(\mathcal{B}_{i_1}, \mathcal{O}) \leq \ldots \leq d(\mathcal{B}_{i_m}, \mathcal{O})$
6: $q_0 \leftarrow$ solution of the stack of tasks $(t_1, \ldots, t_{\text{COUNTER}})$
7: COUNTER $\leftarrow$ COUNTER $+ 1$
8: **end while**
9: **return** q_0

We will now get into the detail of line 4 of algorithm 3, in which we have to test the static stability of a configuration.

D. Testing the static stability

Suppose we have the robot $\mathcal{R}$ in configuration q and we want to check whether or not it is statically stable in this configuration, according to definition 2. In order to get a linear system, we need to consider the modeling of each friction cone $\mathscr{C}_{A_{\mathcal{B}_i},\mathbf{n}_i,\theta_i}$ as discrete *polyhedral cone* with a finite number of generators $\mathbf{u}_{i,1},\dots,\mathbf{u}_{i,n_i}$

$$\begin{aligned}\mathscr{C}_{A_{\mathcal{B}_i},\mathbf{n}_i,\theta_i} &= \mathscr{C}(\mathbf{u}_{i,1},\dots,\mathbf{u}_{i,n_i}) \\ &= \left\{ \sum_{j=1}^{n_i} \lambda_j \mathbf{u}_{i,j} \;/\; \lambda_1,\dots,\lambda_{n_i} \in \mathbb{R}^+ \right\}\end{aligned}$$

which is the set of all *non negative* linear combinations of the generators. With this modeling, we have

$$\mathbf{f}_i \in \mathscr{C}_{A_{\mathcal{B}_i},\mathbf{n}_i,\theta_i} \iff \exists\,(\lambda_{i,j})_{j=1..n_i} \in \left(\mathbb{R}^+\right)^{n_i},\ \mathbf{f}_i = \sum_{j=1}^{n_i} \lambda_{i,j}\mathbf{u}_{i,j}$$

allowing us to rewrite the static stability condition as a linear problem

$$\exists\,(\lambda_{i,j})_{\substack{i\in I(q)\\ j=1..n_i}} \in \prod_{i\in I(q)} \left(\mathbb{R}^+\right)^{n_i},$$
$$\text{s.t.} \begin{cases} \sum_{\substack{i\in I(q)\\ j=1..n_i}} \lambda_{i,j}\mathbf{u}_{i,j} + m\mathbf{g} = \mathbf{0} \\ \sum_{\substack{i\in I(q)\\ j=1..n_i}} \mathscr{M}_O(\lambda_{i,j}\mathbf{u}_{i,j}) + \mathscr{M}_O(m\mathbf{g}) = \mathbf{0} \end{cases}$$

The system of two 3-dimensional equations can be written as a single 6-dimensional equation, putting

$$\mathbf{a}_{i,j} = \begin{pmatrix} \mathbf{u}_{i,j} \\ \mathscr{M}_O(\mathbf{u}_{i,j}) \end{pmatrix} \quad \text{and} \quad \mathbf{v} = -\begin{pmatrix} m\mathbf{g} \\ \mathscr{M}_O(m\mathbf{g}) \end{pmatrix}$$

the static stability condition then becomes

$$\exists\,(\lambda_{i,j})_{\substack{i\in I(q)\\ j=1..n_i}} \in \prod_{i\in I(q)} \left(\mathbb{R}^+\right)^{n_i},\ \text{s.t.} \sum_{\substack{i\in I(q)\\ j=1..n_i}} \lambda_{i,j}\mathbf{a}_{i,j} = \mathbf{v}$$

which can be read as the membership of $\mathbf{v}$ in the cone generated by the $\mathbf{a}_{i,j}$ vectors

$$\mathbf{v} \in \mathscr{C}(\mathbf{a}_{i,j})_{i,j}$$

To solve this system, we used some results that come from the *polyhedral convex cone theory* that we detail hereafter.

Polyhedral convex cone theory: Let $\mathscr{C}(\mathbf{a}_1,\dots,\mathbf{a}_m)$ be the cone generated by $\mathbf{a}_1,\dots,\mathbf{a}_m$ in $\mathbb{R}^n$

$$\mathscr{C}(\mathbf{a}_1,\dots,\mathbf{a}_m) = \left\{ \sum_{j=1}^{m} \lambda_j \mathbf{a}_j \mid \lambda_1,\dots,\lambda_m \in \mathbb{R}^+ \right\}$$

the *dual cone* (also called the *polar cone*) $\mathscr{C}^p$ is defined as

$$\mathscr{C}^p(\mathbf{a}_1,\dots,\mathbf{a}_m) = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \forall i \in \{1,\dots,m\}\ \mathbf{x}^T\mathbf{a}_i \le 0 \right\}$$

Minkowski [13] demonstrated that the polar cone is a cone too, *i.e.* $\exists\ \mathbf{b}_1,\dots,\mathbf{b}_k\ \in \mathbb{R}^n$ such that

$$\mathscr{C}^p(\mathbf{a}_1,\dots,\mathbf{a}_m) = \mathscr{C}(\mathbf{b}_1,\dots,\mathbf{b}_k)$$

The *Farkas lemma* [14] states that $(\mathscr{C}^p)^p = \mathscr{C}$ *i.e.*

$$\mathscr{C}(\mathbf{a}_1,\dots,\mathbf{a}_m) = \mathscr{C}^p(\mathbf{b}_1,\dots,\mathbf{b}_k)$$

this result allows us to test the membership of a vector $\mathbf{x} \in \mathbb{R}^n$ in the dual of the dual cone instead of the cone itself

$$\mathbf{x} \in \mathscr{C}(\mathbf{a}_1,\dots,\mathbf{a}_m) \iff \mathbf{x} \in \mathscr{C}^p(\mathbf{b}_1,\dots,\mathbf{b}_k)$$

or

$$\exists(\lambda_j)_j \in (\mathbb{R}^+)^m\ \mathbf{x} = \sum_{j=1}^{m} \lambda_j\mathbf{a}_j \iff \forall i \in \{1,\dots,k\}\ \mathbf{x}^T\mathbf{b}_i \le 0$$

The second member of this latter equivalence is much easier to check than the first one, if we could compute the vectors $\mathbf{b}_1,\dots,\mathbf{b}_k$. The *Motzkin's double description algorithm* [15] achieves this. We implemented a variation of the original algorithm, proposed by Padberg [16], that allows us to compute a minimal set of generators for the dual cone.

III. Results

We implemented the ideas presented in the previous section within the HPP framework using KineoCAM's software Kineo Path Planner and KineoWorks as a core collision-free motion planning and collision detection module. The model we used for the humanoid robot is HRP-2 [17] which has 36 degrees of freedom (including the free-flyer). The collision-free path planning algorithms we choose are either basic PRM [5] or bidirectional RRT [6].

The main scenario we considered is the highly constrained one demonstrated in [3] which consists in standing up from a chair and going away from a table. The robot is sitting on the chair in initial configuration and is standing by the table at final configuration. The guide obtained is shown in figure 3 while the contacts points plan is illustrated in figure 4. Using the distance to goal as a potential function the robot ends up climbing the table and the contacts planning stops after having consumed all the memory resource of the computer. With the provided guide the contacts planner finds the solution in approximately 3h30min on a standard Pentium IV system, after approximately 10min of computation for the guide.

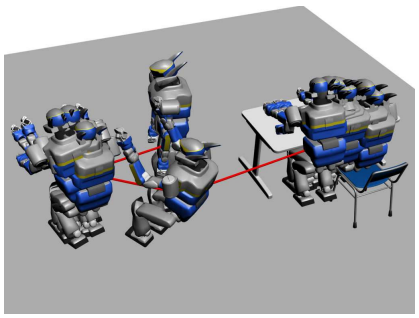


Fig. 3. Guide planning for the out-of-table-and-chair scenario.

We also tested the guide planner on other scenarios on which we have not yet tested the contacts planner, simply to demonstrate the ability of the guide planner of going through different situations (Figs. 5a and 5b).

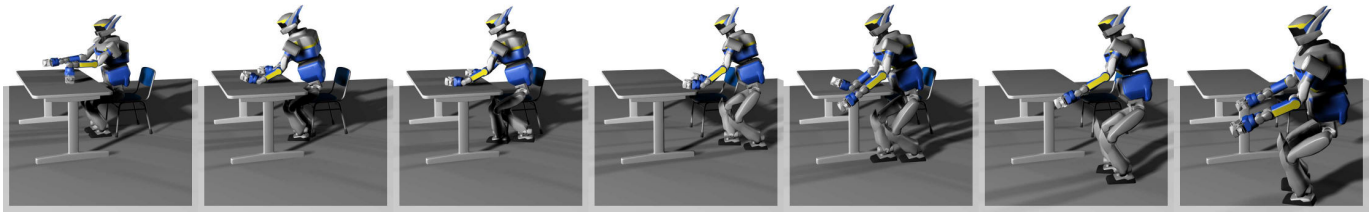
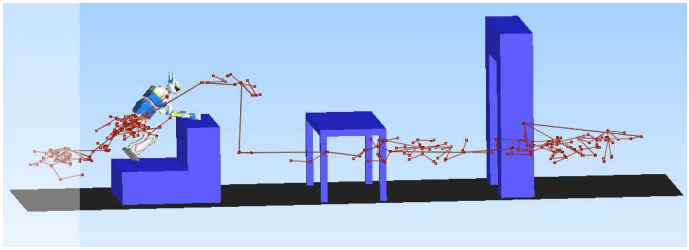
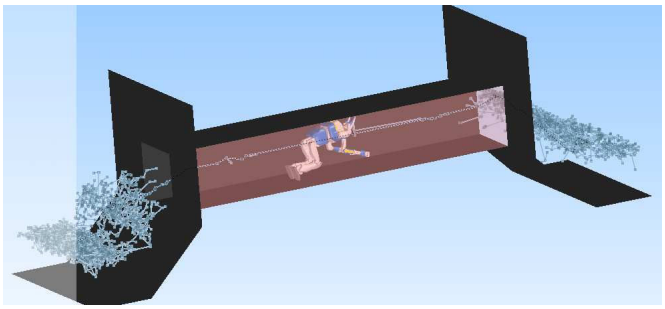


Fig. 4. Contacts planning for the out-of-table-and-chair scenario following the guide provided by the contacts guide planner.



(a) Over the sofa.



(b) Through the tunnel.

Fig. 5. Different scenarios.

Although the gain in computing time that we achieve at the contact-points planner's level is theoretically infinite, computing time at the contacts-guide planner's level remains relatively high for scenarios such as 5a and 5b (a few hours). The time is consumed both on distance computation and stack of tasks solving which are solicited at each iteration of the algorithm.

IV. Conclusion

Improvements of our contacts guide planner are still possible and need to be considered, especially regarding line 7 of algorithm 2. Ensuring that the continuous direct path linking two configurations in guide space lies in the guide space remains an unanswered question in our work. We also still need to work on the linking method that computes the direct path between two guide space's configurations, and which is for now a linear direct path linking method. We added a dimension and thus "volume" to $\mathcal{C}_{\text{guide}}$ in order to pass the test line 7 with higher probability; however, a better solution would be to apply a projection to the whole linear direct path in order to make it fit inside $\mathcal{C}_{\text{guide}}$. These are all questions we plan to investigate in future work.

Acknowledgment

This work is partially supported by grants from the ROBOT@CWE EU CEC project, Contract No. 34002 under the 6th Research program WWW.ROBOT-AT-CWE.EU and by grants from the ImmerSence EU CEC project, Contract No. 27141 WWW.IMMERSENCE.INFO (FET-Presence) under FP6.

References

[1] K. Hauser, T. Bretl, and J.-C. Latombe, "Non-gaited humanoid locomotion planning," in *IEEE-RAS International Conference on Humanoid Robots*, 2005, pp. 7–12.

[2] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on HRP-2," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2974–2979.

[3] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning support contact-points for acyclic motions and experiments on HRP-2," in *International Symposium on Experimental Robotics*, 2008.

[4] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[5] L. E. Kavraki, P. Svetska, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, June 1996.

[6] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 995–1001.

[7] N. M. Amato, O. B. Bayazit, C. L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Workshop on Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[8] K. V. Mardia and P. E. Jupp, *Directional Statistics*. Wiley, 2000.

[9] A. T. A. Wood, "Simulation of the von mises fisher distribution," *Communications in statistics. Simulation and computation*, vol. 23, no. 1, pp. 157–164, 1994.

[10] J. Cortes, T. Simeon, and J. P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using PRM methods," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 2141–2146.

[11] M. Stillman, "Task constrained motion planning in robot joint space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3074–3081.

[12] E. Yoshida, O. Kanoun, C. Esteves-Jaramillo, and J. P. Laumond, "Task-driven support polygon reshaping for humanoids," in *IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 208–213.

[13] H. Minkowski, "Theorie der konvexen korpern, insbesonder der begrundung ihres oberflchenbegriffs," *Gesammelte Abhandlungen*, vol. 2, pp. 131–229, 1911.

[14] J. Farkas, "Uber der einfachen ungleichungen," *Journal fuer die Reine und Angewandte Mathematik*, vol. 124, pp. 1–27, 1902.

[15] T. S. Motzkin, H. Raiffa, G. L. Thomson, and R. M. Thrall, "The double description method," *Contributions to theory of games*, vol. 2, pp. 51–73, 1953.

[16] M. W. Padberg, *Linear Optimization and Extensions*, 2nd ed. Springer, 1999.

[17] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 1083–1090.