Integrating Dynamics into Motion Planning for Humanoid Robots

Fumio Kanehiro, Wael Suleiman, Florent Lamiraux, Eiichi Yoshida and Jean-Paul Laumond

***Abstract*— This paper proposes an whole body motion planning method for humanoid robots in which dynamics is integrated. The method consists of two stages. A collision-free and statically stable *path* is planned in the first stage and it is transformed into a dynamically stable *trajectory* in the second stage. Contributions of the method is summarized as follows. (1) A local method plans a C^1 path while avoiding collisions between non-strictly convex objects. (2) The second stage gives the minimum time trajectory by time parameterization under dynamic balance constraints. (3) Any path reshaping for recovering collision-freeness is not required since the second stage doesn't change shape of the path. Effectiveness of the method is examined by applying it to scenarios of a humanoid robot HRP-2.**

I. INTRODUCTION

An whole body motion planning of a humanoid robot is a challenging problem mainly from the following reasons. (1) Most of humanoid robots have more than 30 degrees of freedom. (2) The whole body motion must satisfy constraints, collision-freeness, physical capabilities and dynamic stability at the same time. Because of these reasons, it is difficult to solve the problem within reasonable time by applying existing motion planning techniques in straightforward way.

Several challenges have already been done on this topic.

[1] proposes a whole-body control framework which provides joint torques by projecting operational tasks into the constraint null-space. We prefer to generate motions in the joint velocity space. Because most of existing humanoid robots are position controlled and it's difficult to control them using joint torques from a practical point of view.

[2] proposes a motion planning method under obstacles and dynamic balance constraints. It consists of two stages. In the first stage, it finds a collision-free path by exploring a set of pre-computed statically stable postures. In the second stage the path is transformed into a dynamically stable trajectory by applying a dynamics filter. Since the dynamics filter modifies shape of the path, it possibly cause collisions. Therefore, the trajectory is confirmed to be collision-free and its duration is extended if collisions are found. [3] also proposes a two stages approach for passing under obstacles. It plans a statically stable path in the first stage like [2] but the path is transformed into a dynamically stable trajectory without changing shape of the path.

The authors are with IS/AIST-ST2I/CNRS Joint Japanese-French Robotics Laboratory(JRL)

F. Kanehiro and E. Yoshida are with Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology(AIST), Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568 Japan {f-kanehiro, e.yoshida}@aist.go.jp

W. Suleiman, F. Lamiraux and J.-P. Laumond are with LAAS-CNRS, university of Toulouse, 7 Avenue du Colonel Roche, 31077 Toulouse, France {suleiman, florent, jpl}@laas.fr

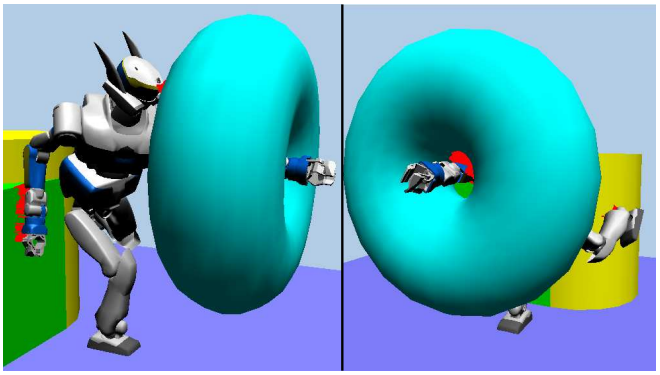

Fig. 1. Whole body reaching with avoiding collisions and keeping stability

[4], [5] and [6] propose methods that integrate motion planning techniques and a bipedal walking pattern generator. The method of [4] is applied to a bar carrying task. It plans a collision-free path at first by approximating a robot by a parallelepiped and then a trajectory is generated by the walking pattern generator. If collisions are caused by dynamic effects, the initial path is reshaped. Methods in [5] and [6] are applied to passing motion under obstacles. [5] regards a robot as a height adjustable box and plans its position, orientation and height. Then the planned path is transformed into a trajectory by a walking pattern generator. [6] generates a walking pattern at first with monitoring collisions and then colliding parts are modified while keeping the horizontal position of the center of mass.

Our strategy is also one of two stages strategies. A collision-free and statically stable *path* is planned in the first stage and it is transformed into a dynamically stable *trajectory* in the second stage. Contributions of the method are summarized as follows. (1) A local method plans a C^1 path while avoiding collisions between non-strictly convex objects. (2) The second stage gives the minimum time trajectory by time parameterization under dynamic balance constraints. (3) Any path reshaping for recovering collision-freeness is not required since the second stage doesn't change shape of the path.

This work is a continuation of [7] and integrates dynamics into a local method in [7].

The paper is organized as follows. In section II, an outline of our method is introduced. In section III, a local method to get a collision-free and statically stable path is explained. In section IV, a timing planning method to transform the path into a minimum time and dynamically stable trajectory is presented. In section V, the effectiveness of the proposed method is confirmed by applying it to two scenarios for a

humanoid robot HRP-2[8]. In section VI, we summarize and conclude the paper.

II. Two Stages Approach

An whole body motion of a humanoid robot must satisfy the following constraints to be feasible for the real robot.

1) There is no self-collision and collision with the environment.
2) All joint angles stay within joint movable ranges.
3) All joint velocities don't exceed their limits.
4) ZMP stays inside of the support polygon.

We propose a method which consists of two stages. The first stage plans a collision-free *path* and it is transformed into a dynamically stable *trajectory* in the second stage like [2]. Here, *path* is a series of configurations and *trajectory* associates time and these configurations. In that sense, *path* is a projected image of *trajectory* onto the configuration space.

If the transformation of the second stage modifies shape of the path, its collision-freeness might be broken. In the case, we need to go back to the first stage and iterate these two stages until a collision-free and dynamically stable trajectory is obtained. In order to prevent this iteration, we plan a statically stable path in the first stage and transform it into a dynamically stable trajectory without changing shape of the path by timing planning. By using a statically stable path as an input of timing planning, the second stage can always obtain a dynamically stable trajectory by slowing down[9].

The timing planning problem is an old problem in robotic research. In the research works on manipulators, the main objective was to reduce the execution time of the tasks, thereby increasing the productivity. Most of these approaches is based on time-optimal control theory (see [10] for an overview). In the framework of mobile robots, the timing planning problem arises also to transform a feasible path to a feasible trajectory [11]. The main objective, in this case, is to reach the goal position as fast as possible.

However, in our case the application of time optimal control theory is a difficult task. This is because not only the dynamic equation of the humanoid robot motion is very complex, but also applying time optimal control theory requires the calculation of the derivative of the configuration space vector of humanoid robot with respect to the parameterized path. Although such a calculation can be evaluated from differential geometry, it is a very difficult task in the case of high dimensional degree of freedoms and branched kinematic chains, which is the case of humanoid robot. For that, we propose to solve the timing planning problem numerically using finite difference approach as it will be explained later in the sequel.

In order to get a short trajectory by timing planning, the first stage plans a C^1 path. If the path is not C^1 at some points, the robot must stop at those points. Because if the robot doesn't stop, discontinuous velocities and infinite joint accelerations are applied to joints and it is impossible to keep dynamic stability. Therefore a C^1 path is preferable to prevent such a "stop-and-go" trajectory is generated.

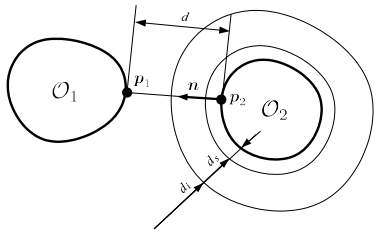

Fig. 2. Collision avoidance: Faverjon and Tournassoud's method

Requirements for a path planned in the first stage is summarized as follows.

1) There is no self-collision and collision with the environment.
2) All joint angles stay within joint movable ranges.
3) The path is composed of statically stable postures.
4) The path is C^1.

These constraints are not broken by timing planning in the second stage. The second stage transforms the path under the following constraints.

5) All joint velocities don't exceed their limits.
6) ZMP stays inside of the support polygon.

III. Collision-Free and Statically Stable Path

A. Local Based Path Planning Method

In order to plan a collision-free and statically stable path, we use a local based path planning method proposed by Faverjon and Tournassoud[12]. The method produces the velocity of the robot under constraints. In the method, a non-constrained initial velocity to achieve a task is computed and this initial guess is projected over the subspace of velocities satisfying linear equality and inequality constraints.

Let $\boldsymbol{q}$ denote the configuration and $\dot{\boldsymbol{q}}$ the velocity of the robot. The projection is formulated as an optimization problem described as follows.

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}} \|\boldsymbol{J}_\tau(\boldsymbol{q})\dot{\boldsymbol{q}} - \underline{\dot{\boldsymbol{\tau}}}\|^2 \\ \text{subject to} \quad &\boldsymbol{A}\dot{\boldsymbol{q}} + \boldsymbol{b} = \boldsymbol{0}, \\ &\boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{d} \leq \boldsymbol{0}. \end{aligned} \tag{1}$$

where $\boldsymbol{J}_\tau(\boldsymbol{q})$ and $\underline{\dot{\boldsymbol{\tau}}}$ denote Jacobian matrix of the task and the desired task velocity respectively. The path under constraints is generated by solving this problem repeatedly until a configuration which satisfies $\boldsymbol{\tau}(\boldsymbol{q}) = \boldsymbol{0}$ is obtained.

B. Constraints

1) Collision Avoidance: A linear inequality constraint to avoid collision is also proposed in [12]. It is called *velocity damper*. Let d be the distance between points, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ on two objects, $\mathcal{O}_1$ and $\mathcal{O}_2$(Fig. 2). When d is smaller than a threshold called *influence distance* and denoted by d_i, *velocity damper* is activated for velocity of d:

$$\dot{d} \geq -\xi \frac{d - d_s}{d_i - d_s} \tag{2}$$

where ξ is a positive coefficient for adjusting convergence speed, $d_s(< d_i)$ is a positive value called *security distance*. d is constrained not to be smaller than d_s. $\dot{d}$ is computed by the following equation.

$$\dot{d} = (\dot{\boldsymbol{p}}_1 - \dot{\boldsymbol{p}}_2 | \boldsymbol{n})$$

where $\boldsymbol{n}$ is the unit vector $(\boldsymbol{p}_1 - \boldsymbol{p}_2)/d$ and notation $(\boldsymbol{u}|\boldsymbol{v})$ refers to the inner product of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. The region where d is smaller than *influence distance* is called *influence zone*. *velocity damper* expresses that d must not decrease too fast when it is smaller than d_i.

The velocity of $\boldsymbol{p}_i (i = 1, 2)$ can be expressed as:

$$\dot{\boldsymbol{p}}_i = \boldsymbol{J}(\boldsymbol{q}, \boldsymbol{p}_i)\dot{\boldsymbol{q}}$$

where $\boldsymbol{J}(\boldsymbol{q}, \boldsymbol{p}_i)$ is a Jacobian matrix of $\mathcal{O}_i$ at $\boldsymbol{p}_i$. Inequality (2) thus becomes a linear inequality constraint over the robot velocity $\dot{\boldsymbol{q}}$:

$$(\dot{\boldsymbol{q}}|\boldsymbol{J}(\boldsymbol{q}, \boldsymbol{p}_1)^T \boldsymbol{n} - \boldsymbol{J}(\boldsymbol{q}, \boldsymbol{p}_2)^T \boldsymbol{n}) \geq -\xi \frac{d - d_s}{d_i - d_s} \qquad (3)$$

If $\mathcal{O}_1$ and $\mathcal{O}_2$ are strictly convex objects, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the closest points on them. In order to use non-strictly convex polyhedra as geometric models of the robot and the environment, several pairs of points must be selected to get continuous velocities as solutions of Problem (1).

The interaction between polyhedra is decomposed into a set of interactions between faces. Polygonal faces are assumed to be decomposed into triangles[1]. The interaction between triangles is managed by the combinatorics between edges of the triangle and Voronoi regions of the other triangle. Let us recall the definition of Voronoi region.

Definition: *Voronoi region $\mathcal{VR}(X)$ for feature X. A Voronoi region associated with a feature X of a triangle is a set of points that are closer to X than any other feature.*

Pairs of points are selected as follows.

Case1 : The edge is in $\mathcal{VR}(\mathcal{F})$
Two pairs, $(\mathcal{V}_i, \mathcal{V}'_i)(i = 1, 2)$ are constrained, where $\mathcal{V}_i (i = 1, 2)$ are end points of the edge. (a, b) denotes a pair of points, a and b and $\mathcal{V}'_i (i = 1, 2)$ denotes a projection of $\mathcal{V}_i$ onto $\mathcal{F}$ along its normal vector.

Case2 : The edge is in $\mathcal{VR}(\mathcal{E})$
Three pairs, $(\mathcal{V}_i, \mathcal{V}'_i)(i = 1, 2, 3)$ are constrained, where $\mathcal{V}'_i (i = 1, 2)$ is a projected point of the end point $\mathcal{V}_i$ onto $\mathcal{E}$. $\mathcal{V}_3$ and $\mathcal{V}'_3$ are the closest points between the edge and $\mathcal{E}$.

Case3 : The edge is in $\mathcal{VR}(\mathcal{V})$
Three pairs, $(\mathcal{V}_i, \mathcal{V})(i = 1, 2, 3)$ are constrained, where $\mathcal{V}_i (i = 1, 2)$ are end points of the edge. $\mathcal{V}_3$ is the closest point between the edge and $\mathcal{V}$.

For more details on continuous constraints generation between polyhedra, please refer [7].

2) Joint Angle Limits: Joint angles can stay within joint movable ranges by constraining those velocities using *velocity damper*. *velocity damper* for joint angles limits are defined as follows.

[1]In the following, *features* of a triangle are the triangular (open)face, the three edges and the three vertices.

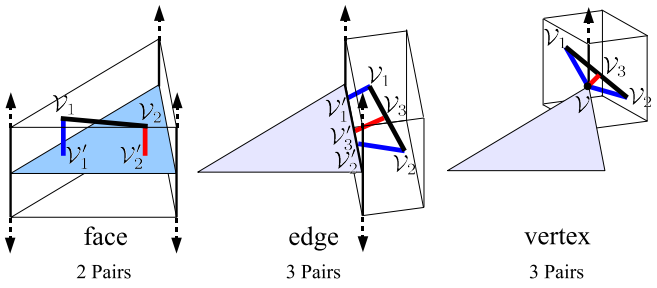

Fig. 3. Constraints generated between an edge and a triangle

$$\dot{q}max_i(q_i) \geq \dot{q}_i \geq \dot{q}min_i(q_i), \text{for } i \in \{1, ..., n\}. \qquad (4)$$

$$\dot{q}max_i(q_i) = \begin{cases} \xi \dfrac{(q_i^+ - q_i) - q_s}{q_i - q_s} & \text{if } q_i^+ - q_i \leq q_i, \\ \dot{q}_i^+ & \text{otherwise} \end{cases} \qquad (5)$$

$$\dot{q}min_i(q_i) = \begin{cases} -\xi \dfrac{(q_i - q_i^-) - q_s}{q_i - q_s} & \text{if } q_i - q_i^- \leq q_i, \\ \dot{q}_i^- & \text{otherwise} \end{cases} \qquad (6)$$

where q_i^+ and q_i^- are physical upper bound and lower bound of joint angle of i^{th} joint respectively.

3) Horizontal Position of the Center of Mass: The robot is statically stable when a projected point of the center of mass(CoM for short) is in the support polygon. CoM is allowed to move as far as the projected point is in the polygon. This can be expressed as a set of inequality constraints. But in order to maximize margin for dynamic effect and minimize the duration of trajectory in the second stage, we constrain the horizontal position of CoM on the line which is orthogonal to the ground. This constraint is expressed as an equality constraint as follows.

$$\boldsymbol{J}_c \dot{\boldsymbol{q}} = \underline{\dot{\boldsymbol{p}}_c} \qquad (7)$$

where $\boldsymbol{J}_c$ and $\underline{\dot{\boldsymbol{p}}_c}$ are Jacobian matrix of CoM and a desired velocity of CoM respectively. $\underline{\dot{\boldsymbol{p}}_c}$ is used to switch supporting legs.

IV. Dynamically Stable Trajectory

A. Dynamic Stability and ZMP

Dynamically stable trajectory is a trajectory for which the ZMP trajectory is always inside of the polygon of support (i,e, the convex hull of all points of contact between the support foot (feet) and the ground). Theoretically, any statically stable trajectory can be transformed into a dynamically stable one by slowing down the humanoid robot's motion.
However, our objective is to find a minimum time and dynamically stable trajectory from a statically stable one. In order to obtain a motion within the humanoid robot capacities, the joint velocity limits of the humanoid robot should be taken into account.

Finally, the timing planning problem can be seen as an optimization problem under inequality constraints.

Let the ZMP on the horizontal ground be given by the following vector

$$\boldsymbol{p} = \begin{bmatrix} p_x & p_y \end{bmatrix}^T \tag{8}$$

To compute $\boldsymbol{p}$, one can use the following formula

$$\boldsymbol{p} = \boldsymbol{N} \ \frac{\boldsymbol{n} \times \boldsymbol{\tau}}{(\boldsymbol{f}|\boldsymbol{n})} \tag{9}$$

where $\boldsymbol{N}$ is a constant matrix

$$\boldsymbol{N} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{10}$$

the vector $\boldsymbol{n}$ is the normal vector on the horizontal ground ($\boldsymbol{n} = \boldsymbol{z}$). The operator $\times$ refers to the cross product. Recall that $(\boldsymbol{f}|\boldsymbol{n})$ is the inner product of the vectors $\boldsymbol{f}$ and $\boldsymbol{n}$.

The vector f is the result of the gravity and inertia forces

$$\boldsymbol{f} = M\boldsymbol{g} - \sum_{i=1}^{n} m_i \ \ddot{\boldsymbol{c}}_i \tag{11}$$

where $\boldsymbol{g}$ denotes the acceleration of the gravity ($\boldsymbol{g} = -g\boldsymbol{z}$), and M is the total mass of the humanoid robot. The quantities m_i, $\ddot{\boldsymbol{c}}_i$ are the mass of the i^{th} link and the acceleration of its center of mass $\boldsymbol{c}_i$ respectively.

Finally, $\boldsymbol{\tau}$ denotes the moment of the force $\boldsymbol{f}$ about the origin of the fixed world frame $\boldsymbol{p}_O$. The expression of $\boldsymbol{\tau}$ is the following

$$\boldsymbol{\tau} = \sum_{i=1}^{n} \left(m_i \ \boldsymbol{c}_i \times (\boldsymbol{g} - \ddot{\boldsymbol{c}}_i) - \dot{\boldsymbol{\mathcal{L}}}_{c_i} \right) \tag{12}$$

where $\boldsymbol{\mathcal{L}}_{c_i}$ is the angular momentum at the point $\boldsymbol{c}_i$

$$\dot{\boldsymbol{\mathcal{L}}}_{c_i} = \boldsymbol{R}_i \left(\boldsymbol{I}_{c_i} \dot{\boldsymbol{\omega}}_i - (\boldsymbol{I}_{c_i} \boldsymbol{\omega}_i) \times \boldsymbol{\omega}_i \right) \tag{13}$$

$\boldsymbol{R}_i$ is the rotation matrix associated to the i^{th} link. $\boldsymbol{I}_{c_i}$, $\boldsymbol{\omega}_i$ and $\dot{\boldsymbol{\omega}}_i$ are its inertia matrix, angular velocity and angular acceleration respectively.

B. Timing planning formulation

Generally speaking, the timing planning problem of a function $f(x_t)$, where t denotes time, consists into finding a real function $\mathcal{S}_t$ in such a way $f(x_{\mathcal{S}_t})$ verifies some temporal constraints, e.g,

$$h(\mathcal{S}_t) \leq f(x_{\mathcal{S}_t}) \leq l(\mathcal{S}_t) \tag{14}$$

Definition: *The spatial velocity vector $\boldsymbol{V}_t \in \mathbb{R}^6$ is defined as follows*

$$\boldsymbol{V}_t = \begin{bmatrix} \boldsymbol{v}_t \\ \\ \boldsymbol{\omega}_t \end{bmatrix} = \begin{bmatrix} \frac{d\boldsymbol{X}_t}{dt} \\ \\ \frac{d\boldsymbol{\theta}_t}{dt} \end{bmatrix} \tag{15}$$

Definition: *The spatial acceleration vector $\dot{\boldsymbol{V}}_t \in \mathbb{R}^6$ is defined as follows*

$$\dot{\boldsymbol{V}}_t = \begin{bmatrix} \boldsymbol{a}_t \\ \\ \dot{\boldsymbol{\omega}}_t \end{bmatrix} = \begin{bmatrix} \frac{d\boldsymbol{v}_t}{dt} \\ \\ \frac{d\boldsymbol{\omega}_t}{dt} \end{bmatrix} \tag{16}$$

In order to obtain a causal motion, the function $\mathcal{S}_t$ should be a strictly increasing function, that means $\frac{d\mathcal{S}_t}{dt} > 0$. Therefore we will express $\mathcal{S}_t$ as the integral of a strictly positive function $s_t > 0$, as follows

$$\mathcal{S}_t = \int_{\tau=0}^{t} s_\tau \ d\tau \tag{17}$$

Let us rewrite the quantities in Eqs (15) and (16) with respect to the mapping function $\mathcal{S}_t$

$$\boldsymbol{V}_{\mathcal{S}_t} = \begin{bmatrix} \frac{d\boldsymbol{X}_t}{d\mathcal{S}_t} \\ \\ \frac{d\boldsymbol{\theta}_t}{d\mathcal{S}_t} \end{bmatrix} \approx \begin{bmatrix} \frac{\Delta \boldsymbol{X}_t}{s_t \ \Delta t} \\ \\ \frac{\Delta \boldsymbol{\theta}_t}{s_t \ \Delta t} \end{bmatrix} \tag{18}$$

$$\begin{aligned} \dot{\boldsymbol{V}}_{\mathcal{S}_t} = \frac{d \ \boldsymbol{V}_{\mathcal{S}_t}}{d \ \mathcal{S}_t} &\approx \frac{1}{\Delta \mathcal{S}_t} \left(\boldsymbol{V}_{\mathcal{S}_t} - \boldsymbol{V}_{\mathcal{S}_{t-1}} \right) \\ &\approx \begin{bmatrix} \frac{\frac{\Delta \boldsymbol{X}_t}{\Delta t} s_{t-1} - \frac{\Delta \boldsymbol{X}_{t-1}}{\Delta t} s_t}{s_t^2 s_{t-1} \Delta t} \\ \\ \frac{\frac{\Delta \boldsymbol{\theta}_t}{\Delta t} s_{t-1} - \frac{\Delta \boldsymbol{\theta}_{t-1}}{\Delta t} s_t}{s_t^2 s_{t-1} \Delta t} \end{bmatrix} \end{aligned} \tag{19}$$

C. Minimum Time and Dynamically Stable Trajectory

Let us suppose that the first stage provides a path which consists of L points. At first, we transform this path to a trajectory by considering a uniform time distribution function. In other words, we suppose that $s_t = 1 : \ \forall t$ in (17). We denote $T = L \ \Delta t$.

In our case, we would obtain a minimum time trajectory which is not only dynamically stable but also it should respect the joint velocity limits of the humanoid robot. Therefore, the optimization problem can be formulated as follow

$$\begin{aligned} \min \ \mathcal{S}_T = \min_{s_t} \int_{t=0}^{T} s_t \ dt \\ \text{subject to} \quad s_t > 0 \\ \boldsymbol{p}_{s_t}^- \leq \boldsymbol{p}_{s_t} \leq \boldsymbol{p}_{s_t}^+ \\ \dot{\boldsymbol{q}}^- \leq \dot{\boldsymbol{q}}_{s_t} \leq \dot{\boldsymbol{q}}^+ \end{aligned} \tag{20}$$

where $\boldsymbol{p}_{s_t}$ is the ZMP vector, $\boldsymbol{p}_{s_t}^-$ and $\boldsymbol{p}_{s_t}^+$ design the polygon of support for the humanoid robot.

The vector $\dot{\boldsymbol{q}}_{\boldsymbol{s_t}}$ denotes the joint velocity of the humanoid robot, and $\dot{\boldsymbol{q}}^-$ and $\dot{\boldsymbol{q}}^+$ design its upper and lower limits.

Let us write $\boldsymbol{p}_{s_t}$ as function of s_t

$$\boldsymbol{p}_{s_t} = \boldsymbol{N} \ \frac{\boldsymbol{n} \times \tilde{\boldsymbol{\tau}}}{(\tilde{\boldsymbol{f}}|\boldsymbol{n})} \tag{21}$$

where

$$\begin{aligned} \tilde{\boldsymbol{\tau}} &= \sum_{i=1}^{n} \left(m_i \ \boldsymbol{X}_t^{c_i} \times \left(\boldsymbol{g} - \tilde{\ddot{\boldsymbol{c}}}_i \right) - \tilde{\dot{\boldsymbol{\mathcal{L}}}}_{c_i} \right) \\ \tilde{\boldsymbol{f}} &= M\boldsymbol{g} - \sum_{i=1}^{n} m_i \ \tilde{\ddot{\boldsymbol{c}}}_i \end{aligned} \tag{22}$$

in which

$$\begin{aligned}
\tilde{\ddot{\boldsymbol{c}}}_i &= \frac{\frac{\Delta \boldsymbol{X}_t^{c_i}}{\Delta t} s_{t-1} - \frac{\Delta \boldsymbol{X}_{t-1}^{c_i}}{\Delta t} s_t}{{s_t}^2 s_{t-1} \Delta t} \\
\tilde{\dot{\boldsymbol{\mathcal{L}}}}_{c_i} &= \boldsymbol{R}_i \left(\boldsymbol{I}_{c_i} \tilde{\dot{\boldsymbol{\omega}}}_i - (\boldsymbol{I}_{c_i} \tilde{\boldsymbol{\omega}}_i) \times \tilde{\boldsymbol{\omega}}_i \right) \\
\tilde{\dot{\boldsymbol{\omega}}}_i &= \frac{\frac{\Delta \boldsymbol{\theta}_t^{c_i}}{\Delta t} s_{t-1} - \frac{\Delta \boldsymbol{\theta}_{t-1}^{c_i}}{\Delta t} s_t}{{s_t}^2 s_{t-1} \Delta t} \\
\tilde{\boldsymbol{\omega}}_i &= \frac{\Delta \boldsymbol{\theta}_t^{c_i}}{s_t \Delta t}.
\end{aligned} \tag{23}$$

In similar way we obtain

$$\dot{\boldsymbol{q}}_{s_t} = \frac{\Delta \boldsymbol{q}_t}{s_t \Delta t} \tag{24}$$

It is clear that the optimization problem (20) is polynomial in s_t, so the gradient and Jacobian functions can be calculated easily.

D. Discretization of solution space

In real fact, the space of the admissible solutions of the minimization problem (20) is very large. In order to transform this space to a smaller dimensional space, we can use a basis of shape functions (e,g cubic B-spline functions). Let us consider a basis of shape functions B_t that is defined as follows

$$\boldsymbol{B}_t = \begin{bmatrix} B_t^1 & B_t^2 & \cdots & B_t^l \end{bmatrix}^T \tag{25}$$

where B_t^i denotes the value of shape function number i at the instant t, the dimension of $\boldsymbol{B}_t$ is l defines the dimension of the basis of shape functions.
The projection of s_t into the basis of shape functions B_t can be given by the following formula

$$s_t = \sum_{i=1}^{l} s_B^i \; B_t^i = \boldsymbol{s}_B^T \; \boldsymbol{B}_t \tag{26}$$

Thus, the optimization problem (20) can be written as follows

$$\begin{aligned}
& \min_{s_B} \sum_{k=1}^{l} s_B^k \int_{t=0}^{T} B_t^k \; dt \\
\text{subject to} \quad & \boldsymbol{s}_B^T \boldsymbol{B}_t > 0 \\
& \boldsymbol{p}_{s_B}^- \leq \boldsymbol{p}_{s_B} \leq \boldsymbol{p}_{s_B}^+ \\
& \dot{\boldsymbol{q}}^- \leq \dot{\boldsymbol{q}}_{s_B} \leq \dot{\boldsymbol{q}}^+
\end{aligned} \tag{27}$$

The optimization problem has been transformed into finding the vector $\boldsymbol{s}_B \in \mathbb{R}^l$, and, in its new form, it can be solved using C-FSQP [13].

Note that the support polygon is a function of $\boldsymbol{s}_B$, and it depends on the horizontal position of CoM. However, as the path provided by the first stage is a statically stable one, the path can be split into various sections. Each section is a statically stable path which has a fixed support polygon and it is independent from $\boldsymbol{s}_B$.

E. Implementation Algorithm of Second Stage

The algorithm of the second stage can be summarized as follows

1) Given a path.
2) Transform the path to a trajectory by considering a uniform time distribution function.
3) Calculate $\Delta \boldsymbol{q}_t$, $\Delta \boldsymbol{X}_t^{c_i}$ and $\Delta \boldsymbol{\theta}_t^{c_i}$ from the obtained trajectory.
4) Choose Δt, e,g, $\Delta t = 5.10^{-3}[s]$.
5) Calculate the cubic B-spline functions.
6) Split the path into verious sections depending on the place and shape of support polygon during the motion. The support polygon for each section is fixed and independent from the time.
7) Solve the optimization problem (27) for each path with the initial solution obtained from the above steps.

V. Simulation

In this section, we confirm effectiveness of our strategy using two examples.

A. Toy World Scenario

The first example is a collision-free reaching motion in the cluttered environment. In this example, the task of the robot is to move its left hand to the specified position $\underline{\boldsymbol{p}_h}$. The objective function of Problem (1) is given as follows.

$$\| \boldsymbol{J}_h \dot{\boldsymbol{q}} - \underline{\dot{\boldsymbol{p}}_h} \|^2$$

where

$$\underline{\dot{\boldsymbol{p}}_h} = \delta p_{max} \frac{\underline{\boldsymbol{p}_h} - \boldsymbol{p}_h(\boldsymbol{q})}{\| \underline{\boldsymbol{p}_h} - \boldsymbol{p}_h(\boldsymbol{q}) \|} \tag{28}$$

Inequality (3),(4) and Equality (7) are used as constraints over joint velocities. ξ, d_i and d_s of Inequality (3) are set to 0.5[m/s], 0.05[m] and 0.03[m] respectively. Those of Inequality (4) are set to 0.5[rad/s], 0.2[rad] and 0.05[rad] respectively. In this example, since the robot is standing on the right leg from the beginning, $\underline{\dot{\boldsymbol{p}}_c}$ is set to $\boldsymbol{0}$.

The reaching motion is obtained by solving Problem (1) until the hand reaches $\boldsymbol{p}_g$. Figure 4 shows the initial configuration and the final one. Figure 1 also shows the final configuration from another viewpoint. Red lines show pairs of points to be constrained. The robot is standing on its right leg with surrounded by a torus, a cylinder and a box in the initial configuration. The reaching target is placed at the other side of the torus. In the final configuration, the left arm and the head are avoiding collisions with the torus and the left leg and the right arm are with the cylinder and the box respectively.

The computational time of the first stage is mainly decided by the number of constraints for collision avoidance and it depends on the number of pairs of triangles in *influence zone*. The computational time can be saved by reducing the number of pairs. The reduction can be done by the following two methods. The first one is to make *influence zone* thin by making the difference between d_i and d_s small. But it leads to big accelerations since the velocity of constrained

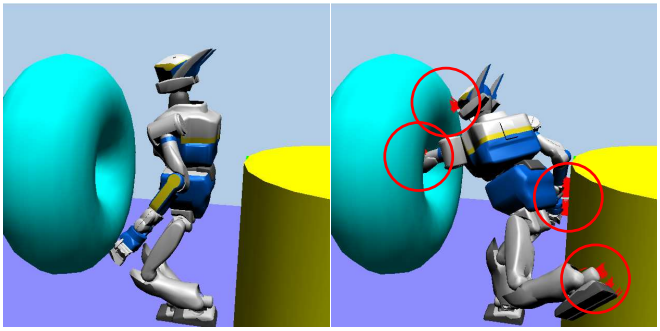

Fig. 4. Initial and final configurations of whole body reaching: In the final configuration, the left arm and the head are avoiding collisions with the torus and the left leg and the right arm are with the cylinder and the box respectively.

points are decreased quickly. This is not preferable to obtain a short trajectory in the next stage. The second method is to simplify geometric models of the robot and the environment. If the simplified model contains the precise one in it, the path obtained using simplified model is also valid for the precise one.

Figure 5 shows a precise model and a simplified one of HRP-2. The precise one consists of 20858 triangles and the simplified one does of 1140 triangles. The computational times and the number of constraints are compared while combining these models and models of torus. The precise torus consists of 800 triangles and the simplified one does of 288 triangles. Table I shows averages of the computational time in [s] and the number of constraints (the number between parentheses) for one computational step. The computational time can be drastically reduced by using a simplified model.

Once the collision-free path is available, this path is converted to a trajectory using a uniform time distribution function. The obtained trajectory is then used to initialize Problem (27). Solving Problem (27) yields a minimum time and dynamically stable trajectory. The computation time of the second stage is obviously depends on the length of the path and the dimension of the basis of cubic B-spline functions. For this scenario, we have chosen: $\Delta t = 5.10^{-3}[s]$, and a basis of 120 cubic B-spline functions. The function s_t is given in Figure 6. The duration of minimum time dynamically stable trajectory is around $24[s]$ instead of $77[s]$ for the original uniform time distribution trajectory. That means the optimized trajectory is around 3 times faster than the original one. However it's not always the case, for example if the original trajectory is fast and dynamically unstable, then the optimized trajectory will be slower. Though the optimized trajectory is the minimum time trajectory of the dynamically stable trajectories. The trajectory of ZMP is given in Figure 7.

B. In a Room Scenario

The second example is on HRP-2 in its room. Figure 8 shows snapshots of this scenario. Each row shows snapshots along time-line and each column does from different viewpoints. In the first frame, HRP-2 is sleeping but there is no

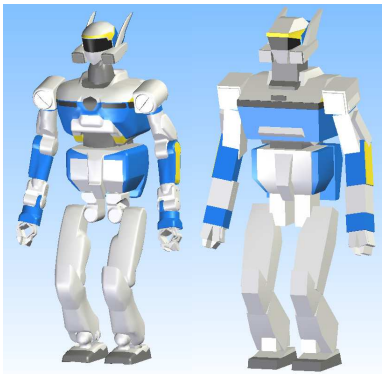

Fig. 5. Precise model(left) and simplified model(right)

TABLE I

AVERAGES OF THE COMPUTATIONAL TIME[S] AND THE NUMBER OF CONSTRAINTS

		torus	
		precise	simplified
HRP-2	precise	1.47(3460)	0.45(1803)
	simplified	0.10(698)	0.05(375)

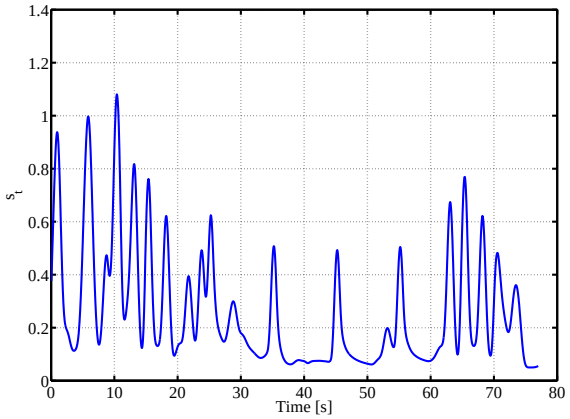

Fig. 6. Time parameterization function (s_t)

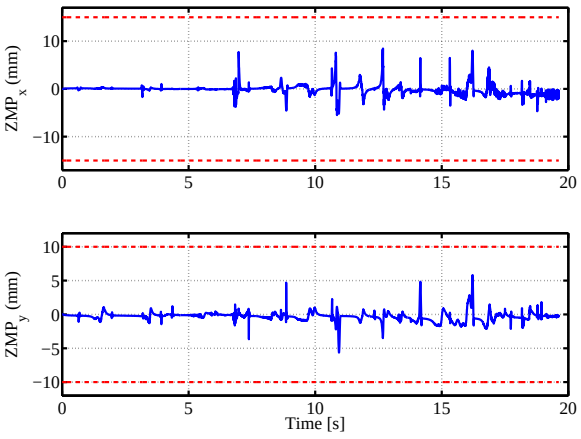

Fig. 7. ZMP_x and ZMP_y trajectories in solid lines, and the safety zones for dynamical stability are designed by the dashed lines

contacts among HRP-2, the chair and the table. We inserted gaps which are bigger than d_s since constraints for collision avoidance don't work correctly if distances are smaller than d_s. And then it stands up(b), gets out of a gap between the table and the chair(c–e) and tries to grasp something on the shelf(f).

The scenario is split into three parts, (A) stretching, (B) stepping and (C) reaching to switch objective functions and constraints in Problem (1). In part (A) the task is to going to a goal configuration $\underline{\boldsymbol{q}}$ and the objective function is defined as follows.

$$\|\dot{\boldsymbol{q}} - \underline{\dot{\boldsymbol{q}}}\|^2$$

where

$$\underline{\dot{\boldsymbol{q}}} = \delta q_{max} \frac{\underline{\boldsymbol{q}} - \boldsymbol{q}}{\max_{i=1,\cdots,n} |\underline{q_i} - q_i|}$$

The part (B) is split into two parts again, (B-1) moving CoM and (B-2) moving foot. Multiple steps can be done by repeating these two parts. Since the first stage must generate a statically stable path, a resultant walking is a static one. In part (B-1), $\underline{\dot{\boldsymbol{p}}_c}$ is computed in the same way with Eq.(28). Paths for the part (B-2) and (C) are generated in the same way with the previous example. Goal configurations and points of feet and the hand are selected by hand.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a collision-free and dynamically stable whole body motion planning method. The method and its contributions are summarized as follows.

- The method consists of two stages. A collision-free statically stable path is planned in the first stage. The path is transformed into a minimum time and dynamically stable trajectory in the second stage.
- A local method in the first stage can generate a differentiable path while avoiding collisions between non-strictly convex objects.
- Since the first stage plans a statically stable path and the second stage doesn't change shape of the path, the trajectory can be always obtained without iterating two stages.

Since the path is generated using a local method, the planner might be trapped by local minima. We are woking on integration with a global method like PRM[14] or RRT[15].

The followings are limitations of our strategy and future works.

- Since the first stage of the method must plan a statically stable path, this method can't be applied to faster motions where CoM goes out of the support polygon like a biped walking.
- The robot might slip or jump. Because forces and the moment around the vertical axis are not constrained.

VII. ACKNOWLEDGMENTS

We gratefully acknowledge AEM Design, developer of a powerful QP solver, C-FSQP[13].

REFERENCES

[1] L. Sentis and O. Khatib, "A Whole-Body Control Framework for Humanoids Operating in Human Environments," in *Proc. of International Conference on Robotics and Automation*, 2006, pp. 2641–2648.

[2] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion Planning for Humanoid Robots Under Obstacle and Dynamic Balance Constraints," in *Proc. of the 2001 IEEE International Conference on Robotics & Automation*, 2001, pp. 692–698.

[3] H. Sanada, E. Yoshida, and K. Yokoi, "Passing Under Obstacles with Humanoid Robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS'07)*, 2007, pp. 4028–4034.

[4] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond, "Humanoid Motion Planning for Dynamic Tasks," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2005, pp. 1–6.

[5] E. Yoshida, "Humanoid Motion Planning using Multi-Level DOF Exploitation based on Randomized Method," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS'05)*, 2005, pp. 25–30.

[6] K. Harada, S. Hattori, H. Hirukawa, M. Morisawa, S. Kajita, and E. Yoshida, "Motion Planning for Walking Pattern Generation of Humanoid Robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS'07)*, 2007, pp. 4227–4233.

[7] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A Local Collision Avoidance Method for Non-strictly Convex Objects," in *Technical report LAAS-CNRS 08025*, 2008.

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

[9] I. Belousov, C. Estevès, J.-P. Laumond, and E. Ferré, "Motion planning for the large space manipulators with complicated dynamics," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS'05)*, 2005, pp. 3713–3719.

[10] M. Renaud and J. Y. Fourquet, "Time-optimal Motions of Robot Manipulators Including Dynamics," *The robotics review 2*, pp. 225–259, 1992.

[11] F. Lamiraux and J.-P. Laumond, "From Paths to Trajectories for Multi-body Mobile Robots," in *Proc. of the Fifth International Symposium on Experimental Robotics*, London, UK, 1998, pp. 301–309.

[12] B. Faverjon and P. Tournassoud, "A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom," in *Proc. of IEEE International Conference on Robotics and Automation*, 1987, pp. 1152–1159.

[13] C. Lawrence, J. L. Zhou, and A. L. Tits, *User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints.*

[14] L. E. Kavraki, P. Svestka, 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, August 1996.

[15] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *TR98-11, Computer Science Dept., Iowa State University*, October 1998.

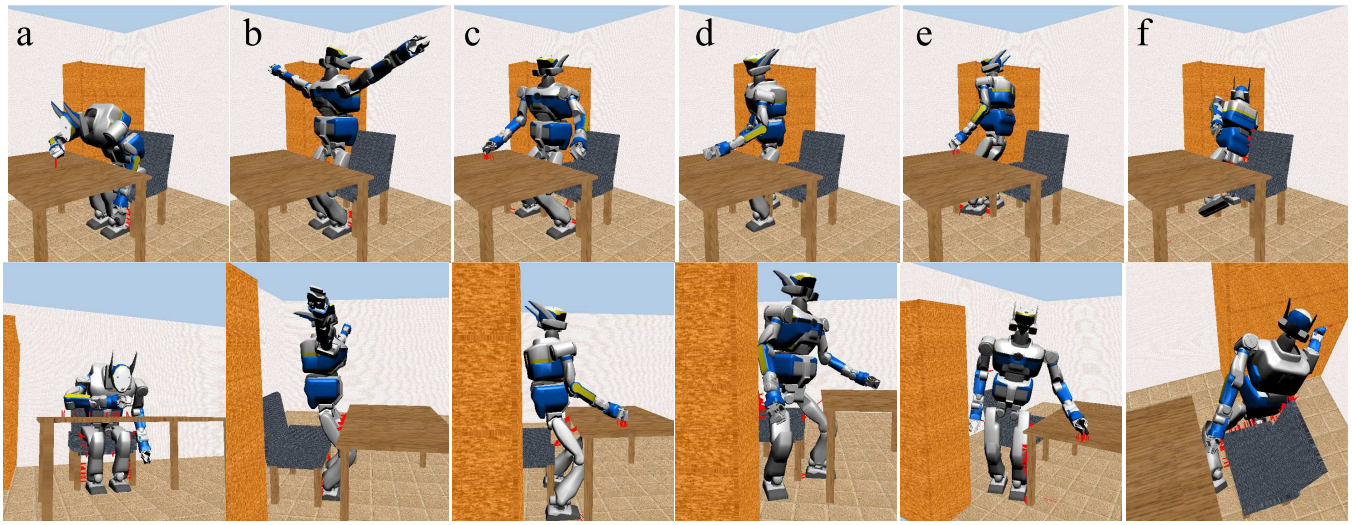

Fig. 8. Snapshots of in a room scenario