Continuous gradient proximity distance for humanoids free-collision optimized-postures

Adrien Escande, Sylvain Miossec and Abderrahmane Kheddar
AIST/CNRS Joint Japanese-French Robotics Laboratory (JRL)

***Abstract*— This paper presents a new method for strictly convex hulls (i.e. bounding volume) generation made by assembling patches of spheres and toruses (STP-BV). This bounding volume allows to compute proximity distances with the guarantee of their gradients' continuity. This bounding volume is computed off-line; it slightly over-covers the polyhedral convex hull of the geometrical form. Given a pair of convex objects, having only one of them strictly convex (i.e. STP-BV covered) is proved to be sufficient to guarantee gradient continuity of the proximity distance. The distance computation is based on the closest features of the underlying polyhedral convex hull obtained with V-Clip or any other algorithm. The suggested algorithm is exemplified through a free-collision (including free self-collision) optimization-based humanoid posture generation.**

I. INTRODUCTION

This work is motivated from the following problem: in [1], a planner for humanoid acyclic motion is proposed; it is composed mainly of two interrelated modules: (i) a stance tree explorer and (ii) a posture generation modules. The stance for each step of the planner is obtained from an optimization-based posture generation using C-FSQP [2]. Optimal trajectories in robotics can be computed by solving an optimization problem on a cost function. This function (generally involving minimum energy consumption, speed or precision, etc.) is defined together with a set of constraints that encompasses joint limits, contacts and path tracking in the Cartesian space, stability... see [3] [4] and their inside bibliography for more details on robotics trajectory optimization and optimal control.

By using FSQP, or any other available optimization software, collision avoidance can be integrated as a constraint among others. Indeed, one may consider writing these constraints using any available proximity distance algorithm which returns signed proximity distances separating two bodies; see the recent exhaustive books [5], [6]. Not having collision between two bodies is equivalent to keep the separating distance always positive. Consequently, using self-collision and collision avoidance constraints in optimization software does not appear really problematic (again not considering the implementation issues which may reveal to be not that simple).

However, most optimization softwares require the gradients of the criteria and the constraints to be continuous with respect to the parameters (here robot joints and eventually trajectories' parameters) -and even the Hessian. The proximity distance between polyhedrons does not meet such a requirement. Yet, there exist optimization algorithms not requiring continuous gradients, such as the Bundle methods [7]. But they are not as fast as methods requiring continuous gradients. Improving their convergence properties are still open research problems.

The main motivation of this work is to guarantee that computed gradients of proximity distance are continuous function of the parameters in order to use fast optimization methods. The problem of ensuring continuous distance's gradient has, to the best knowledge of the authors, never been treated before and the proposed method is new. Continuity properties of the distance have been merely discussed or even assumed in previous works. For example, in [8] [9], this problem has been addressed in a 2D case, it has been claimed that the distance between convex objects is smooth and thus the gradient is continuous. The latter assertion is not always valid unless one object is strictly convex, the former depends on the continuity properties of both objects' surfaces, as we will demonstrate in section II. The flaw in the demonstration is to suppose that witness points of the distance are smooth functions of the parameters. Assumption about the witness points' continuity is always implicitly made in papers computing distance's gradient whereas the strict convexity of at least one object is not addressed. It is only in [10] that the non differentiability (and non-convexity) of the distance between convex bodies is well addressed and used with non-smooth analysis in the context of sensory-based planning. Our approach, on the contrary, draws solution to get rid of the non-differentiability: we build off-line strictly convex bounding volume that can be considered as a smooth 'rounding' of the polyhedron convex hull. Detailed theoretical aspects of this work are described in [**?**]. The distances are computed on the basis of a proximity distance algorithm V-Clip [11]. The distance computation needs little additional time than the V-Clip distance computation. We exemplified it in a collision-free (including self-collision) static posture generator for the humanoid robot HRP-2.

II. PROXIMITY DISTANCE CONTINUITY

A. Problem definition and notations

In this section we consider the distance between two convex objects O_1 and O_2. This distance will be denoted δ. The relative position between the two objects is parameterized by the vector q having 6 scalars (3 for the rotation, 3 for the translation). δ is a function of q.

We call witness points a pair of points of $O_1 \times O_2$ that are at the distance δ. Under certain conditions (see theorem 2.1), this pair is unique for a given relative position. Thus we can define $p_{\min}(q) = (p_{1\,\min}(q), p_{2\,\min}(q))^T$ the function that associates the position of witness points to each q.

Additionally, the surface of each object can be described by a function of two parameters (for example, in spherical coordinates). Let u be the 4-dimensional vector of these two times two parameters, and r_1 and r_2 these functions for O_1 and O_2 (we define as function from a subset of $\mathbb{R}^4$ to $\mathbb{R}^3$ to simplify the writing even though both of them are functions from a subset of $\mathbb{R}^2$ to $\mathbb{R}^3$).
Witness points being on the objects' surfaces, we define $u_{\min}$ as the function of q that returns the vector u of these points. Denoting by $R(q)$ the rotation matrix parametrized by q, and by $T(q)$ the translation vector, we have $p_{\min}(q) = \left(r_1\left(u_{\min}(q)\right), R(q)r_2\left(u_{\min}(q)\right) + T(q)\right)^T$
Finally, we define $f(q,u) = r_1(u) - R(q)r_2(u) + T(q)$ the distance vector between the two points parametrized by u,

$$\delta(q) = \min_u \|f(q,u))\| = \|f\left(q, u_{\min}(q)\right)\| \quad (1)$$

B. Strict convexity of a body

Intuitively, a gradient discontinuity occurs when there is a jump between witness points pairs. This is the case around a configuration for which there is a non unique witness pair, such as when an edge is parallel to a face. When objects are not in collision, non uniqueness of the witness pair is directly linked to the non strict convexity of the objects: considering the computation of distance as a minimization problem, this problem is not strictly convex if both objects are not strictly convex. Global minimum may thus be reached in several points. We thus need at least one of the objects to be strictly convex, while the other may be only convex.

Theorem 2.1: (Unicity of witness points) There is a unique pair of witness points if at least one of the bodies is strictly convex.

Theorem 2.2: (Main result) The witness points of the minimum distance between two convex bodies are continuous functions of q if at least one of the bodies is strictly convex ($u_{\min}$ and $p_{\min}$ are continuous functions of q).

Proof: First, if no body is strictly convex, witness points are not necessarily unique thus we have no continuity. We only need to demonstrate that strict convexity implies continuity. The idea driving the demonstration of witness points continuity consists in building some small volumes that include old and new witness points of the minimum distance, and to show that the volumes tend to points when the infinitesimal transformation tends to zero.

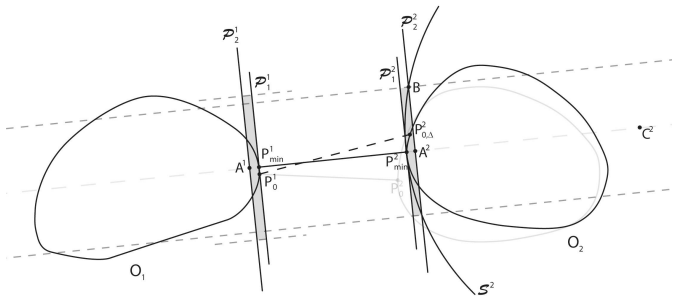

Fig. 1. Demonstration of the continuity of $P_{\min}$.

Let's consider two convex objects O_1 and O_2, the latter being strictly convex. Because of this strict convexity, spheres exist that completely include O_2 while being tangent to it, whatever the tangent point. Let R be the radius of one of this spheres.
We consider the objects for a relative position described by q_0. P_0^1 and P_0^2, respectively on O_1 and O_2 are the unique witness points for this position. $p_{\min}(q_0) = (P_0^1, P_0^2)^T$
Let's move to the position $q_0 + \Delta q$. We note $P^2_{0,\Delta}$ the new position of the point P_0^2 and $P^1_{\min}$ and $P^2_{\min}$ the new witness points so that $p_{\min}(q_0 + \Delta q) = (P^1_{\min}, P^2_{\min})^T$.
Since δ satisfy a Lipschitz condition, there is a real K such as $|\delta(q_0 + \Delta q) - \delta(q_0)| \leq K\Delta q$, see also [10].
Let $\mathcal{P}_1^1$ and $\mathcal{P}_1^1$ be respectively the tangent planes to O_1 and O_2 in $P^1_{\min}$ and $P^2_{\min}$. $\mathcal{P}_2^2$ (resp. $\mathcal{P}_2^1$) is the plane parallel to $\mathcal{P}_1^2$ (resp. $\mathcal{P}_1^1$) distant of $\delta(q_0) + K\Delta q$ to $P^1_{\min}$ (resp. $P^2_{\min}$). $\mathcal{S}^2$ is the sphere tangent to O_2 in $P^2_{\min}$ with a radius R. Its center C^2 is aligned with $P^1_{\min}$ and $P^2_{\min}$. A^1 and A^2 are the points of $\mathcal{P}_2^1$ and $\mathcal{P}_2^2$ on this alignment and B is one point on the intersection of $\mathcal{P}_2^2$ with $\mathcal{S}^2$ (this intersection is not void as soon as Δq is small enough).

Let $\sigma = \sqrt{|\delta^2(q_0 + \Delta q) - \delta^2(q_0)|}$. σ tends to 0 when Δq do so.
We define $\mathcal{C}^1_{\Delta q}$ (resp. $\mathcal{C}^2_{\Delta q}$) as the cylinder of axis $\left(P^1_{\min}P^2_{\min}\right)$ and radius $A^2B + \sigma$ (resp. A^2B) delimited by $\mathcal{P}_1^1$ and $\mathcal{P}_2^1$ (resp. $\mathcal{P}_1^2$ and $\mathcal{P}_2^2$). $A^2B + \sigma$ is the maximal distance between $P^1_{\min}$ and P_0^1, obtained for the extreme case where $P^2_{0,\Delta}$ is on the boundary of $\mathcal{C}^2_{\Delta q}$.
Let $E_{\Delta q} = \mathcal{C}^1_{\Delta q} \times \mathcal{C}^2_{\Delta q}$.
By construction $\left(P^1_{\min}, P^2_{\min}\right)$ and $\left(P_0^1, P^2_{0,\Delta q}\right)$ are in $E_{\Delta q}$.
$A^1P^1_{\min} = A^2P^2_{\min} = \delta(q_0) + K\Delta q - \delta(q_0 + \Delta q)$,
$A^2C^2 = R - A^2P^2_{\min}$,
and $A^2B = \sqrt{(R^{2^2} - A^2C^{2^2})}$.
Since δ is a continuous function, $A^2P^2_{\min}$ and $A^1P^1_{\min}$ tend to 0 when Δq tends to 0. Thus, A^2C^2 tends to R and A^2B tends to 0.
Both cylinders tend to a single point: $E_{\Delta q}$ tends to $\left\{\left(P_0^1, P^2_{0,\Delta q=0}\right)\right\} = \left\{\left(P_0^1, P_0^2\right)\right\}$.
We then have $p_{\min}(q_0 + \Delta q) = \left(P^1_{\min}, P^2_{\min}\right)^T$ tends to $p_{\min}(q_0) = \left(P_0^1, P_0^2\right)$ i.e. the continuity in q_0. ■

Considering the differentiation of $\|f\left(q, u_{\min}(q)\right)\|$ with respect to q, the following result is then almost straightforward:

Theorem 2.3: The minimum distance between two convex bodies is a C^1 function of q if and only if one of the bodies is strictly convex.

If the surfaces of these objects have additional continuity properties, the distance will benefit of it, as shown by the following theorems:

Theorem 2.4: If the surface of both bodies are C^k, with $k \geq 2$ then the witness points are C^{k-1} function of q.

Proof: Let u_0 be the coordinates of the witness points at q_0.
We have $\left(\frac{\partial f}{\partial u}(q_0, u_0)\right)^T f(q_0, u_0) = 0$ (optimality condition)

which can be rewritten $\frac{\partial f^2}{\partial u}(q_0, u_0) = 0$.
For a given q_0, f^2 is the square distance between two points of the bodies' surfaces, and thus is a strictly convex function, which implies $\frac{\partial^2 f^2}{\partial u^2} \neq 0$.
Let's note $F(q, u) = \frac{\partial f^2}{\partial u}(q, u)$. F is C^{k-1}, $F(q_0, u_0) = 0$ and $\frac{\partial F}{\partial u}(q_0, u_0) \neq 0$. Thus u is locally a C^{k-1} function of q (implicit functions theorem). This yields that $u_{\min}$ is a C^{k-1} function of q. ■

Considering again the differentiation of $\|f\left(q, u_{\min}(q)\right)\|$, the following theorem holds:

Theorem 2.5: If the surfaces of both bodies are C^k, with $k \geq 2$ then the minimum distance between them is C^k.

C. Penetration case

The C^n property for $n > 0$ cannot be reached everywhere in the penetration case: the distance minimization problem when penetration occurs is not convex. In some configurations there are several pairs of witness points; jumps between witness pairs are thus inevitable. We can however keep the results of the above theorems for a subset of penetration cases. But first, we need to ensure the continuity properties between the penetration and non-penetration case: as long as an object is not totally included in the other one, we define the penetration distance as the opposite of the distance between the pair of points verifying the optimality condition $\left(\frac{\partial f}{\partial u}\right)^T f = 0$ while being at the minimal distance among the possible pairs (which is ultimately the same definition as in the non penetration case). Under the assumption of "slight" penetration the previous results hold; otherwise, we may then encounter gradient discontinuities, but it has to be pointed out that configurations where these discontinuities occur are repulsive: following the gradient make us going away from them.

III. Sphere-Torus-Patch Bounding Volumes

As shown in the previous section, distance discontinuities arise when there are flat areas in both objects, which is often the case since most of the applications deal with polyhedrons, whose edges and faces are not strictly convex. It is thus needed to round these parts off, while staying close to the original object in a conservative way.
In this article, we propose a way to build a close bounding volume on a polyhedron to make it strictly convex. To each type of feature of the polyhedron we associate our feature:

- each vertex is paired with a small sphere of radius r centered on it,
- each face is covered by part of a big sphere of radius R that is tangent to the spheres of the 3 vertices,
- each edge is associated to a part of torus whose inner radius is R and that connects to the 2 big spheres of the adjacent faces in a C^1 way.

We call *sphere-torus-patche bounding volume* the obtained object and denote it STP-BV. r is the minimal distance between the polyhedron and its STP-BV, it is a security margin. R controls the maximal curvature of the STP-BV as well as the maximal margin. It must be at least the radius of the polyhedron but should be several order bigger for a better approximation of the polyhedron.

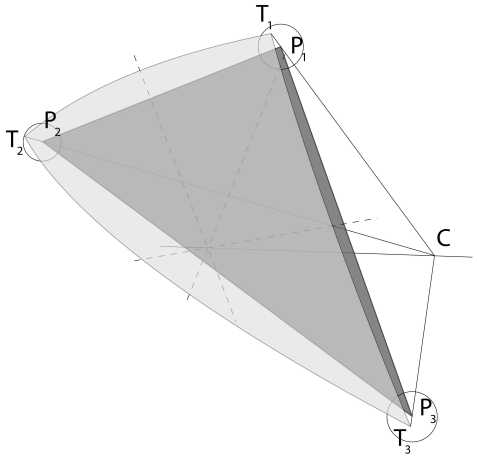

Fig. 2. Sphere construction.

A. Big spheres construction

We consider a triangular face of the polyhedron (fig. 2). P_1, P_2 and P_3 are its vertices given counterclockwise around the outer normal vector. T_1, T_2 and T_3 are the corresponding points where the small spheres are tangent to the big one. C is the center of the big sphere.
Because of the tangency, C, P_i and T_i are aligned ($i = 1, 2, 3$). The problem is reduced to the finding of a sphere of radius $R - r$ that goes through the three vertices of the face, and is *above* the face (direction is given by the outer normal). There is a unique sphere corresponding to this problem.

B. Toruses construction

We obtain the torus above one edge by rotating the sphere of a neighbouring face around this edge and keeping the resulting inner volume.

Theorem 3.1: The distance between the center of sphere corresponding to a face, and the median point of one of its edges depends only of the length l of the edge and is $\sqrt{(R-r)^2 - \frac{l^2}{4}}$

Proof: It is the direct result of the Pythagorean theorem written for the triangle made of the center of the sphere, the median point of the edge and one of its end points. ■

The centers C_1 and C_2 of the two spheres corresponding to the two neighbouring faces of the edge are thus on a same circle centered on the edge middle I and of radius $\sqrt{(R-r)^2 - \frac{l^2}{4}}$.
We consider the circle $\mathcal{C}_1$ of center C_1 and radius R in the plane defined by C_1 and the edge. By construction, this circle coincides with the sphere centered in C_1. Similarly we construct $\mathcal{C}_2$ with center C_2.
By making $\mathcal{C}_1$ revolve around the edge until it coincides with $\mathcal{C}_2$, we obtain the part of torus we need. It should be noticed

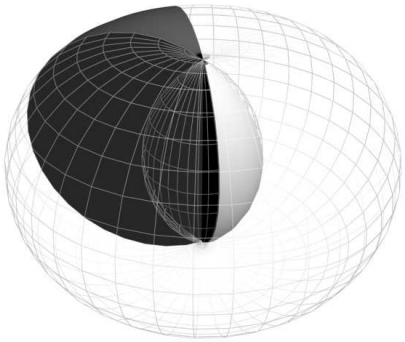

Fig. 3. Torus part. The black and white parts are the portion of torus we consider, the white surface is the part used in the STP-BV

that the torus is not be seen as the usual donut since its usual small radius is bigger than the (usual) big one. The part we consider here is on the inner side of the whole torus as shown in white on the fig 3.

The torus and the spheres coincide in the limit planes and are perpendicular to those planes. The torus is therefore tangent to both spheres and the junction between the torus and one sphere is then C^1 and the resulting volume is strictly convex.

C. Properties

- In the STP-BV, the toruses are tangent to the small spheres.
- The STP-BV is C^1 and strictly convex.
- STP-BVs are even piecewise C^∞ since toruses and spheres are C^∞ surfaces.
- If a is the length of the longest edge of the polyhedron, the maximal margin between the convex hull of the polyhedron and its STP-BV is $R - \sqrt{(R-r)^2 - \frac{a^2}{12}}$

If R is notably bigger than r and a, the expression can be accurately approximated (Taylor expansion) simply by r.
For example, with the values we typically use in our applications ($a = 10$cm, $r = 1$cm and $R = 10$m), the maximal margin is 1.00417cm.

D. Voronoi region

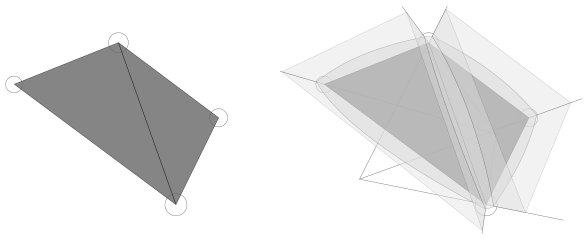

Fig. 5. Voronoi regions around an edge.

Illustrations are given with figures 4 and 5.

An edge is the revolution axis of both its associated torus and the spheres of its vertices. Therefore finding the voronoi regions in a plane containing this edge is enough to determine what the limits of these regions are in 3D. In such a plane, a torus and an adjacent small sphere become two C^1-connected tangent circles. The centers of these two circles and the tangency point are on a same line that is also the boundary of the voronoi region. With the revolution around the edge, we obtain that the limit between the voronoi regions of a torus and a small sphere is thus a cone.

The intersection of any plane $\mathcal{P}_e$ perpendicular to an edge with the big sphere of one of its neighbouring faces and its associated torus is C^1-connected tangent circles. Center of the first circle is the projection of the center of the big sphere onto $\mathcal{P}_e$, center of the other circle is the intersection of the edge with $\mathcal{P}_e$ (by construction). Let us notice three particular possible $\mathcal{P}_e$: the one going through the center of the sphere and the two others passing through each extremity of the edge. As before, the line going through the centers of the two circles is the limit between the voronoi regions of these circles. Therefore the limit between the regions of a torus and a big sphere is a plane defined by the center of the big sphere and the two extremities of the edge.

Between a big sphere and a small one, there is a single common point. Separation of the voronoi regions is the line defined by the centers of both spheres. This line is also part of all the limits between neighbouring voronoi regions.

IV. Computing Proximity Distances and Gradients

The main idea to compute the distance between two polyhedral objects O_1 and O_2 is to rely on a classical distance computation algorithm from which we can retrieve the witness features (closest pair of features of the two objects) and the witness points. We simply add a layer on this algorithm that associates the closest features of the STP-BVs to these witness.

A. Bounding volume construction

The construction of the bounding volume is depicted in algorithm 1. It relies only on the vertices of the object. The main idea is similar to the *gift wrapping algorithm*: we find a first face whose associated sphere contains all the points of the cloud and their associated small spheres. We then make this sphere rotate around the edges of this face until it becomes tangent to the small sphere of a vertex. The edge and the vertex form a new face. Its sphere is the only one containing all the points of the cloud and while being based on the edge, but for the previous sphere. We then rotate around the edges of this new face, until we reach a face already computed.

Finding the first face is made by the *init* function. Since we can only rotate around a single edge at a time, it is needed to have the list of edge around which we haven't yet rotated. We select the edge around which the smallest rotation will be required.

The function *angleMin* takes such an edge structure as input and returns the rotation angle around this edge, as well as the vertex for which this angle is reached. The edge list, called *edgeStack* in the algorithm pseudo-code, thus contains edge structures, each of which is paired with a vertex and sorted

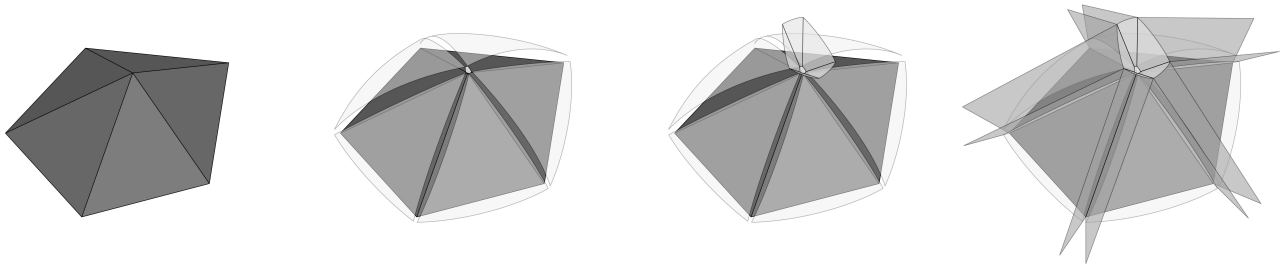

Fig. 4. Voronoi regions around a vertex.

Algorithm: Bounding volume construction: Pseudo-code

Data: cloud of points, value of r and R
Result: set of faces with their spheres
-e, e_1 and e_2: *are edges along with an additional vertex*
-v,v_1,v_2 and v_3: *are a vertices*
-s and s': *are spheres data*
-$vertices$: *the input set of vertices*
-edgeStack: *a list of edges along with two vertices, sorted according to an angle.*
-output: *a list of triangles and their tangent spheres*

```
BuildVolume()
begin
    init(edgeStack, output)
    while (!empty (edgeStack)) do
        (e, v) ← first(edgeStack)
        (e1, e2) ← newEdges(e,v)
        push(output, face(e,v))
        if contains(edgeStack, e1) then
            delete(edgeStack, e1)
        else
            insert(edgeStack, e1, angleMin(e1))
        end
        if contains(edgeStack, e2) then
            delete(edgeStack, e2)
        else
            insert(edgeStack, e2, angleMin(e2))
        end
    end
    return output
end
```

Algorithm 1: Bounding volume construction.

according to the value of its associated angle. Updating this list is the main task of *buildVolume*.

From an edge and a vertex, defining a face, *newEdges* simply builds two new edge structures which corresponds to the two egdes of this face, that contain the vertex. If one of these edges is already in *edgeStack*, we are coming back to an existing face, since the edge has already been created. In this case, the two neighbouring faces of this edge have already been found and the edge must therefore be removed from the list. In the opposite case, it must be inserted in the list. An edge thus appears exactly twice (once for each neighbouring face): it is built a first time and is later used as a rotation axis or is built again. When there is no edge left in the list to be processed, the algorithm terminates.

sphere returns the sphere of a face described either by three vertices or by an edge and a vertex, *face* returns a face along with its sphere for the same input, and *angle* computes the angle between two spheres given an edge.

The obtained hull is not the convex hull, for some points of it may have been ignored because of the curvature of the spheres. However, when r tends to 0 and R becomes infinite, this hull tends to the convex hull of the cloud of points.

B. Overall algorithm

Computing the distance for two (non necessarily convex) polyhedra O_1 and O_2 is done in two steps (fig 6).

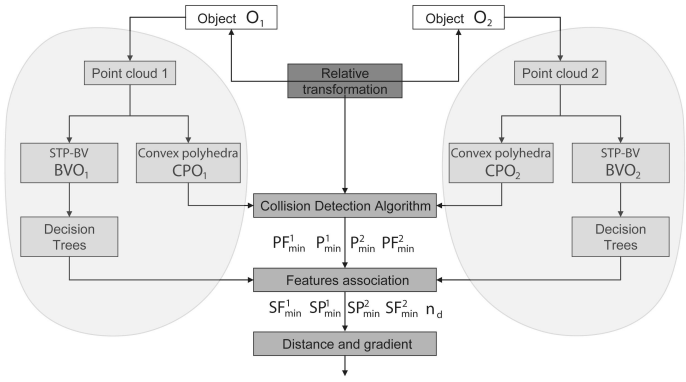

Fig. 6. Overall algorithm. The steps in the gray area are computed off-line.

First an off-line computation produces the two STP-BV BVO_1 and BVO_2, as well as the underlying convex polyhedron CPO_1 and CPO_2 and some data related to the voronoi regions of the STP-BV. This data is aimed at precomputing all that is possible so that the on-line distance computation is as fast as possible. The second step is this on-line computation: we first run a classical collision detection algorithm on CPO_1 and CPO_2 that returns the witness points $P^1_{\min}$ and $P^2_{\min}$ as well as the closest (polyhedral) features PF_1 and PF_2. From this output we then have to find the closest smooth features SF_1 and SF_2 of BVO_1 and BVO_2. Once these features are obtained we are able to compute the distance $\delta = d(SF_1, SF_2)$, the new witness points $SP^1_{\min}$ and $SP^2_{\min}$, and n_d the normal unit vector to BV_1 in $SP^1_{\min}$, the three

latter data being needed for gradient computation. We need to know how to find them for three kinds of pairs of smooth features: sphere-sphere, sphere-torus and torus-torus.

Associating smooth features to polyhedral features is based on two heuristics:

- for a polyhedral feature the smooth feature is to be found among its corresponding smooth features and the latters' direct neighbours,
- the choice of the smooth feature SF_i is based on the position of $P^j_{\min}$ regarding the voronoi region of SF^0_i, the smooth feature directly linked to PF_i ($i = 1$ and $j = 2$ or the contrary).

As shown in V-Clip [11], closest features are reached when the witness point of each object is inside the voronoi region of the closest feature of the other object. This property should apply to $SP^1_{\min}$, $SP^2_{\min}$, SF_1 and SF_2. Tests with objects such as used in the examples of section VI show that such is not the case in 0.3% of the computation requests. In more than 99% of these "failed" cases, the witness point is in a neighbouring smooth feature's voronoi region so a single test is enough to correct the mischoice.

C. Associating a smooth feature to a vertex

The smooth voronoi region of a small sphere always lies strictly inside the polyhedral voronoi region of the associated vertex. If a polyhedral witness point is given to be in the voronoi region of a vertex, it can thus be either in the smooth voronoi region of the associated small sphere or in one of the adjacent smooth voronoi regions (toruses or big spheres of the edges and faces that contain the vertex). The association

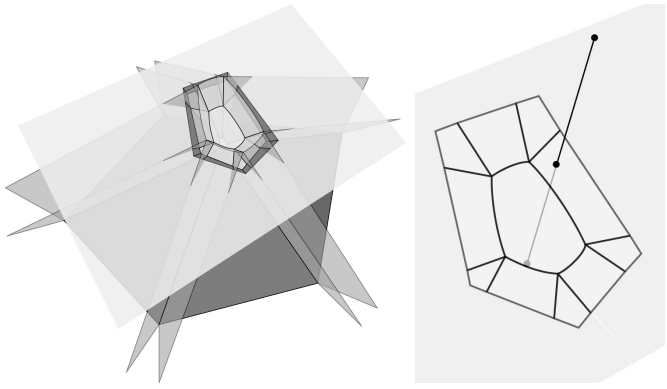

Fig. 7. Intersection of a plane with the voronoi regions related to a vertex.

computation is based on the following remark: the vertex belongs to every surrounding smooth voronoi region limits as well as to its own voronoi cone. Therefore, this vertex can be considered as the focal point of a projection onto a plane: a plane is then chosen above the vertex, whose normal vector is inside the voronoi region of the small sphere. Projection of the smooth voronoi regions onto this plane is the same as the intersection of the voronoi regions with the plane as shown in fig. 7. The rightmost picture of this figure shows the obtain 2D regions in which the witness point of the other object is projected. The projection of this point lies inside the the outer polygone since it is in the polygonal voronoi region of the vertex. Finding which 2D region the projection of the witness point lies in is strictly equivalent to finding which smooth voronoi region the witness point is in, but it requires less calculations.

D. Associating a smooth feature to an edge or a face

Because the smooth voronoi region of a small sphere is always strictly inside the polyhedral voronoi region of the corresponding vertex, the polyhedral voronoi regions of the edges and faces never intersect with it. But smooth and polyhedral voronoi regions of big spheres/faces can intersect in various ways with the regions of toruses/edges, depending on the shapes of the faces. In all cases, we only tested on which sides of the smooth voronoi limit planes the witness point is.

E. Computing Proximity Distances' Gradients

Gradient computation has already been studied. We follow the scheme exposed in [12]: with our previous notation, we have $\frac{\partial \delta}{\partial q}(q) = n_d^T \left(\frac{\partial SP^1_{\min}}{\partial q}(q) - \frac{\partial SP^2_{\min}}{\partial q}(q) \right)$. The optimality condition then yields that the relative motion of the smooth witness points on the boundary surfaces is orthogonal to the normal unit vector n_d so that the expression becomes simpler:

$$\frac{\partial \delta}{\partial q}(q) = n_d^T \left(\frac{\partial SP^1_{\min \in BVO_1}}{\partial q}(q) - \frac{\partial SP^2_{\min \in BVO_2}}{\partial q}(q) \right) \quad (2)$$

The two last derivatives correspond to the velocities of the points that match with the witness points at q and are fixed to the objects. However, comparing to [12], our normal unit vector n_d is derived directly from the smooth features.

For a point P of fixed coordinates (x, y, z) in the local frame of an object O at the configuration(q), the gradient has the following expression: $\frac{\partial P}{\partial q}(q) = xJ_1(q) + yJ_2q(q) + zJ_3(q) + J_4$ obtained by deriving $P(q) = R(q)(x, y, z)^T + T(q) = xC_1(q) + yC_2(q) + zC_3(q) + T(q)$ where R is a rotation matrix, C_i its columns and T is the translation vector. The J_i are the gradient matrices of the C_i and T. These matrices can be analytically computed beforehand and are called hereafter *pre-gradient matrices*.

V. Implementation

We use V-Clip since it meets the requirements of returning witness points and features. However, V-Clip does not perfectly handle the penetration case because it stops to the first intersecting pairs of features. This is in most cases enough to handle "slight" penetrations, but we added a heuristic to handle some deeper ones.

A. Sphere-torus and torus-torus distances

Computing sphere-torus and torus-torus distances reduces to 3D point-circle and circle-circle distance computations respectively. The former has a simple geometrical solution, the latter has been proved in [13] to have no analytical one. Effective and accurate computation of circle-circle distance

has been presented in [14]. Point-circle distance can also be found in this paper as a sub-problem of the circle-circle computation. Since we use an inner part of the toruses, we need to compute the maximum distance with arcs. For the point-circle distance, the farthest point is the opposite of the nearest one so that there are few changes we made. For the circle-circle distance, we adapted [14] to find a maximum instead of a minimum, but we opted for an iterative method.

B. Computation time

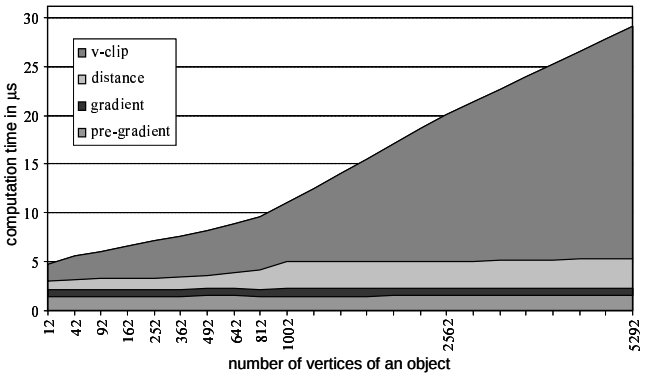

Fig. 8. Computation time for large relative movements

Computation time was recorded for several objects' sizes and two extents of relative movements. Distance is computed between two identical geodesic spheres of radius 20cm covered with STP-BV whose parameters are $r = 1$cm and $R = 10$m. The size of an object is measured by the number of its vertices. For each object size, one million calls were made to the algorithm. Between two calls, both objects rotated around their three axis, and the relative distance was changed, with a total range of 40cm and so that there can be slight penetrations. For the graph in fig. 8, angles are about 20 degrees and the average distance change is 2cm. Angle increments are taken so that the same configuration never appears twice. Computations are made on a 3.4GHz Pentium Xeon with 2GB of RAM.

VI. Application: Free-collision Humanoid Posture Generation

Now that the method is explained, and packaged into a C++ code, we will demonstrate it in a humanoid context. The posture generator proposed in the planner described in [1] is now improved by integrating this method to obtain optimized collision-free postures for a humanoid robot HRP-2. This posture generator is an optimization under constraints program: constraints are physical and geometrical, such as stability, required body positions (robot-environment contacts for example), and collisions. The criterion to be minimized can be any user-defined smooth function. In the following scenarios we use a very simple one that gives fairly human-like postures for upright positions:

$$f(q) = \sum_{i=1}^{n_j} \left(q_i - \frac{q_i^{\min} + q_i^{\max}}{4} \right)^2 \qquad (3)$$

where n_j is the number of joints, and $q_i^{\min}$ and $q_i^{\max}$ are the joint limits of q_i. $\frac{q_i^{\min}+q_i^{\max}}{2}$ being the middle of the joint limits, $\frac{q_i^{\min}+q_i^{\max}}{4}$ is the middle between this middle and 0, the configuration with all angles to 0 being the one in figure 9.

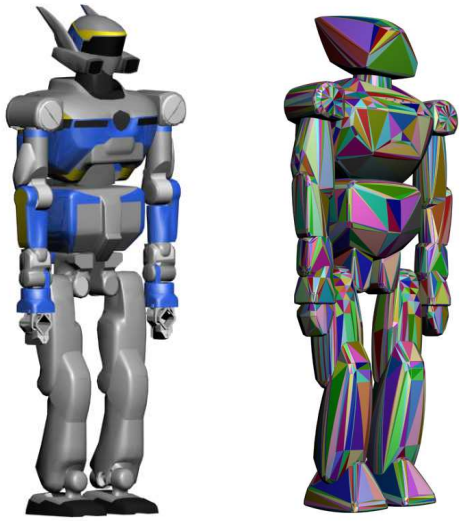

Fig. 9. HRP-2 robot and its STP-BV

Figure 9 shows both the geometrical model of humanoid HRP-2 and its corresponding STP-BV that is computed off-line prior to its use in the experimental scenarios. The model of each HRP-2 body contains between 50 to 800 vertices. Parameters of STP-BV are $r = 1$cm (safe collision margin) and $R = 10$m (chosen to have few difference between safe margin and maximal margin).

Concerning auto-collision, robot pairs that need to be selected for checking have been studied in [15] and recently in [16]. In the latter, the use of look-up tables was proposed to deal with composed joints, so that the safety margins of bounding volumes does not restrain the movement possibilities for this kind of joint. We also focused on that point; however, since we needed to have and compute continuous gradients for all constraints, such a method was not possible, and we had to use specific analytical functions to prevent collision around the hip, waist, neck and shoulder joints. These functions were obtained either geometrically or by experimentations on a real HRP-2 robot. The proximity distances between each pair of bodies must be positive (or above a given threshold, which is anyway already taken into account by the very nature approximation of the STP-BV). The gradients of these constraints are computed by the proposed method. We also define obstacles to be avoided and describe body-obstacle pairs to be checked in the same way we did for auto-collision.
There are 117 auto-collision constraints; 8 of them are analytical constraints, the other being computed with the STP-BVs.

Pick-can-from-fridge scenario: the robot is asked to grab a can in a fridge. For that it is constrained to have its two feet on the floor and its left hand around the can, as shown in the leftmost picture in fig. 10. Collisions with the environment

Fig. 10. Pick-can-from-fridge scenario. From left to right: illustration of the target, posture obtained without collision avoidance constraints, successful free-collision posture found (side and up views).

are checked between bodies of the robots and: the fridge doors, the fridge left panel, the shelves, the carafe and the clementines in front of the can. 33 pairs are involved. The robot's initial posture is its 0 posture, in front of the fridge In the second picture, collisions are not checked: there are collisions between the left knee and the lower door, the left forearm and the carafe, the chest and the inner side of the upper door, the arm and the upper door. The next two pictures show the posture obtained when collisions are checked. Here, the robot is stretched and close to many of its joint limits because of collisions. It results in a narrow feasible space. The computation time is thus quite big: for this scenario it is 0.469 seconds; the distance function is called 11,675 times and the gradient 9,855 times (without collision checking, the posture is computed in 32 milliseconds). The result is obtained in 74 iterations of FSQP. The same posture for a human character should be obtained much faster thanks to a bigger number of DoFs; HRP-2 lacking wrist DoF is a serious limitation here.

VII. Conclusion

A new method for computing proximity distances with continuous gradient is proposed. The main idea is to ensure strict convexity of the bounding envelope that is computed off-line. We suggest to use near-convex hulls built with spheres and toruses patches. The assembly is made in such a way as to at least guarantee C^1. For the time being, the distance computation is based on the closest features of the underlying polyhedral convex hull using V-Clip. The presented algorithm has been successfully exemplified through a collision-free (including self-collision) optimization-based humanoid posture generation for HRP-2. Future work will investigate (i) inclusion of the method as part of the constraints in the low level robot controller: this has been done actually, and will be published in a forthcoming paper, (ii) V-Clip is used as an intermediary step for computing the proximity distance between a pair of bodies. We are now working on a new distance algorithm which get rid of this step.

References

[1] 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 Robots and Intelligent Systems*, Beijing, China, October 2006, pp. 2974–2979.

[2] 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." [Online]. Available: citeseer.ist.psu.edu/341929.html

[3] S.-H. Lee, J. Kim, F. C. Park, M. Kim, and J. E. Bobrow, "Newton-type algorithms for dynamics-based robot movement optimization," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 657– 667, August 2005.

[4] S. Miossec, K. Yokoi, and A. Kheddar, "Development of a software for motion optimization of robots– application to the kick motion of the HRP-2 robot," in *IEEE International Conference on Robotics and Biomimetics*, 2006.

[5] G. van den Bergen, *Collision detection in interactive 3D environments*, ser. The Morgan Kaufmann Series in Interactive 3D Technology, D. H. Eberly, Ed. Morgan Kaufmann Publishers, 2004.

[6] C. Ericson, *Real-time collision detection*, ser. The Morgan Kaufmann Series in Interactive 3D Technology, D. H. Eberly, Ed. Morgan Kaufmann Publishers, 2005.

[7] F. Bonnans, C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical optimization- Theoretical and Practical Aspects*. Springer, September 2002.

[8] J. Y. Lee and H. Choset, "Sensor-based construction of a retract-like structure for a planar rod robot," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 435–449, August 2001.

[9] H. Choset, B. Mirtich, and J. Burdick, "Sensor based planning for a planar rod robot: Incremental construction of the planar Rod-HGVG," in *IEEE International Conference on Robotics and Automation*, vol. 4, April 1997, pp. 3427 – 3434.

[10] S. Rusaw, "Sensor-based motion planning in SE(2) and SE(3) via nonsmooth analysis," Oxford University Computing Laboratory, Tech. Rep., 27 September 2001. [Online]. Available: http://citeseer.ist.psu.edu/476565.html;ftp://ftp.comlab.ox.ac.uk/pub/Documents/techreports/RR-01-13.ps.gz

[11] B. Mirtich, "V-Clip: fast and robust polyhedral collision detection," *ACM Transactions on Graphics*, vol. 17, no. 3, pp. 177–208, 1998.

[12] O. Lefebvre, F. Lamiraux, and D. Bonnafous, "Fast computation of robot-obstacle interactions in nonholonomic trajectory deformation," in *International Conference on Robotics and Automation*, Barcelona, Spain, April 2005. [Online]. Available: http://www.laas.fr/~olefebvr/Publications/icra05.pdf

[13] C. A. Neff, "Finding the distance between two circles in three-dimensional space," *IBM J. Res. Dev.*, vol. 34, no. 5, pp. 770–775, 1990.

[14] D. Vranek, "Fast and accurate circle-circle and circle-line 3d distance computation," *J. Graph. Tools*, vol. 7, no. 1, pp. 23–32, 2002.

[15] J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue, "Self-collision detection and prevention for humanoid robots," in *IEEE International Conference on Robotics and Automation*, Washington DC, May 2002, pp. 2265–2270.

[16] K. Okada and M. Inaba, "A hybrid approach to practical self collision detection system of humanoid robot," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, 2006, pp. 3952–3957.