Strictly convex hulls for computing continuous gradient proximity distances

Adrien Escande, Sylvain Miossec, and Abderrahmane Kheddar, *Member, IEEE*

***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.**

***Index Terms*— Continuous gradients of proximity distances, strictly convex hulls, sphere-torus patches, bounding volume, free collision posture generation.**

I. INTRODUCTION

THE presented work has been motivated from the following problem: in [1], a planner for humanoid acyclic motion is proposed. This planner is composed mainly of two interrelated modules: (i) a stance tree explorer module and (ii) a posture generation module. The stance for each step of the planner is obtained from an optimization-based posture generation using C-FSQP [2]. In a wider context, optimal trajectories in robotics can be computed from 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, etc. Interested readers may refer to [3] [4] and their inside bibliography sections for more details on trajectory optimization and optimal control theory applied in robotics. In these two quoted papers however, and in most we have read, constraints such as robot self-collision and non desired collisions with the environments have not been taken into account. In other works, they are not taken into account deeply unless the context or the robot structure is very specific.

In fact, by using FSQP, or any other available optimization software, collision avoidance might be easily integrated as a constraint among others, nonetheless providing additional software development efforts. Indeed, one may consider writing these constraints using any available proximity distance algorithm which returns signed proximity distances separating two bodies (such as V-Clip[1] or SOLID[2]); the interested reader may refer to the recent exhaustive and excellent review books [5], [6]. 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). Not having collision between two bodies is equivalent to keep the separating distance always positive. Hence, the matter of evaluating the proximity constraints is merely the execution of the proximity computation between the considered pairs of bodies (if the two bodies belong to the same robot, it is called self-collision avoidance).

However, a problem arises with the very fact that most optimization software 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[3]. 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 relatively fast 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. Therefore, a method has been devised and implemented to ensure C^1 property of the proximity distance. This results in a better convergence of the optimization software. Although, the motivation of this work originates from a robotic context, the proposed method extends to all problems where proximity distances requiring continuous gradient is crucial enough to be considered (for instance in control theory). 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 totally new and original. Continuity properties of the distance have been merely discussed or even assumed in previous works. For example, in the work proposed in [8] [9], where 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

Manuscript received June xx, 2007; revised Xxxxxxx 00, 200X. This work was supported by XX.

A. Escande, S. Miossec and A. Kheddar are with the JRL-CNRS, Japan.

[1] `http://www.merl.com/projects/vclip/`

[2] `http://www.dtecta.com/`

[3] In almost all optimization software, approximation of the Hessian appears to be robust enough.

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.

This paper is organized as follows: first, evidence of proximity distances' gradient discontinuity is presented by means of a very detailed example. A set of theorems proving some general properties about the distance continuity is explained, they are to be used in the proposed solution. This is followed by section III where a new method which builds off-line a strictly convex hull (i.e. a strictly convex bounding volume) from 3D point clouds is described along with its construction steps. Section IV deals with how distances are computed on the basis of a proximity distance algorithm such as V-Clip [11], and is followed by a short section V describing how the gradient of the proximity distance is computed. Some implementation details are described in section VI. The paper ends by exemplifying the proposed method 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 6 scalars (3 for the rotation, 3 for the translation). q is the vector made of these scalars.
δ is a function of q and we will study its continuity.
We call witness points a pair of points of $O_1 \times O_2$ that are at the distance δ. Under certain conditions that we will expose after (see 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 pair 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 $p_{\min}(q) = (r_1(u_{\min}(q)), R(q)r_2(u_{\min}(q)) + T(q))^T$
Finally, we define $f(q,u) = r_1(u) - R(q)r_2(u) + T(q)$ the distance vector between the the two points parametrized by u, so that

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

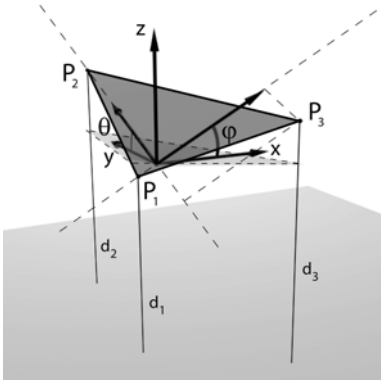


Fig. 1. A triangle above a plane for a position (θ, φ). Dashed triangle is the triangle at the position $(\theta, \varphi) = (0, 0)$

B. Evidence of gradient discontinuities

A planar example is given in [10]. Let us have a look at the distance between a triangle $\mathcal{T}$, part of a given polyhedral mesh, and a plane $\mathcal{P}$ in 3D, figure 1. We consider the initial position where the triangle is above the plane and parallel to it, and we attach an orthogonal frame $\mathcal{F}$ to $\mathcal{T}$ with its z-axis collinear to $\mathcal{P}$'s normal vector. Each point $P_i, i \in \{1,2,3\}$ of the triangle thus has the coordinates $(x_i, y_i, 0)$, and the equation of $\mathcal{P}$ is $z = -z_0$. For the sake of clarity and without loss of generality, we can assume that $x_1 < x_2 < x_3$ and that $\forall i \neq j \in \{1,2,3\}$, $y_i \neq y_j$.
$\mathcal{T}$ rotates around the x-axis (resp. y-axis) with the angle θ (resp. φ). When θ = φ = 0, $\mathcal{T}$ is parallel to $\mathcal{P}$. First rotation is around x-axis
Let us then define three (signed) distances $d_i(\theta, \varphi) = d(P_i, \mathcal{T})$ We have

$$\begin{aligned} d_i &= -x_i \sin\varphi \ + \ y_i \sin\theta\cos\varphi \ - \ z_0 \\ \forall\ \theta,\varphi,\ \delta &= d(\mathcal{T},\mathcal{P}) = \min_{i\in\{1,2,3\}} \ d_i \end{aligned} \tag{2}$$

We also define $D_i = d_i - d_{i+1}$ (subscribts are considered modulo 3). For example, we have

$$\delta = d_1 \ \Leftrightarrow \ d_1 \leq d_2 \text{ and } d_1 \leq d_3 \ \Leftrightarrow \ D_1 \leq 0 \text{ and } D_3 \geq 0$$

It is sufficient to show the discontinuity by restricting $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ and $-\frac{\pi}{2} < \varphi < \frac{\pi}{2}$, which moreover is the case since $\mathcal{T}$ is part of a polyhedron and therefore the inside triangle will not be considered for distance computation. In particular, we thus have $\cos\varphi > 0$
We use j instead of $i+1 \mod 3$ for convenience.
$\forall\ i \in \{1,2,3\}$,

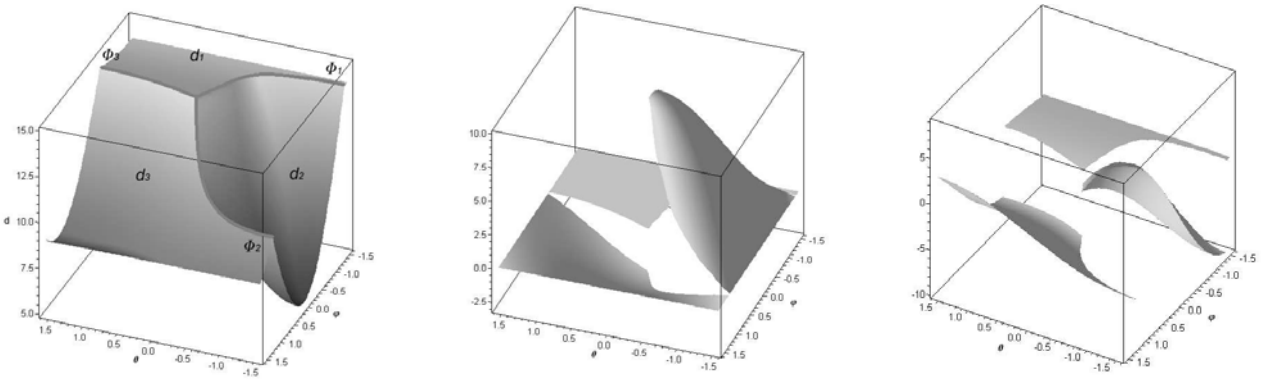
$$\begin{aligned} D_i < 0 \quad &\Leftrightarrow \quad (y_i - y_j)\sin\theta\cos\varphi \ - \ (x_i - x_j)\sin\varphi \ < \ 0 \\ &\Leftrightarrow \quad (y_i - y_j)\sin\theta\cos\varphi \ < \ (x_i - x_j)\sin\varphi \end{aligned} \tag{3}$$

If $i < j$ ($i = 1$ or 2), we have $x_i < x_j$ and then

$$(3) \quad \Leftrightarrow \quad \left(\frac{y_i - y_j}{x_i - x_j}\right)\sin\theta \ > \ \tan\varphi$$

If $i > j$ ($i = 3$), we have $x_i > x_j$ and then

$$(3) \quad \Leftrightarrow \quad \left(\frac{y_i - y_j}{x_i - x_j}\right)\sin\theta \ < \ \tan\varphi$$



(a) distance function δ (b) $\frac{\partial \delta}{\partial \theta}$ (c) $\frac{\partial \delta}{\partial \varphi}$

Fig. 2. Proximity distance function and its gradients corresponding to the figure 1.

Defining $\Phi_i(\theta) = \arctan\left(\frac{y_i - y_j}{x_i - x_j}\sin\theta\right)$ we then have

$$\begin{aligned} \text{if } i<j, \quad D_i<0 \quad &\Leftrightarrow \quad \varphi < \Phi_i(\theta) \\ \text{if } i>j, \quad D_i<0 \quad &\Leftrightarrow \quad \varphi > \Phi_i(\theta) \end{aligned}$$

The same way, we obtain

$$\begin{aligned} \text{if } i<j, \quad D_i>0 \quad &\Leftrightarrow \quad \varphi > \Phi_i(\theta) \\ \text{if } i>j, \quad D_i>0 \quad &\Leftrightarrow \quad \varphi < \Phi_i(\theta) \\ \text{and} \quad D_i=0 \quad &\Leftrightarrow \quad \varphi = \Phi_i(\theta) \end{aligned}$$

D_1	-	-	-	-	+	+	+	+
D_2	-	-	+	+	-	-	+	+
D_3	-	+	-	+	-	+	-	+
δ	/	d_1	d_3	d_1	d_2	d_2	d_3	/

To summarize:

$$\begin{aligned} \delta = d_1 \quad &\Leftrightarrow \quad \varphi \le \Phi_1(\theta) \text{ and } \varphi \le \Phi_3(\theta) \\ \delta = d_2 \quad &\Leftrightarrow \quad \varphi \le \Phi_2(\theta) \text{ and } \varphi \ge \Phi_1(\theta) \\ \delta = d_3 \quad &\Leftrightarrow \quad \varphi \ge \Phi_3(\theta) \text{ and } \varphi \ge \Phi_2(\theta) \end{aligned}$$

The d_i functions are smooth, so that δ is smooth everywhere in $]-\frac{\pi}{2},\frac{\pi}{2}[\times]-\frac{\pi}{2},\frac{\pi}{2}[$ but when δ changes from one d_i to an other, a discontinuity of the gradient always arises. This situation corresponds to:

- $\varphi = \theta = 0$,
- $\varphi = \Phi_1(\theta)$ and $\varphi < \Phi_3(\theta)$ $(\Rightarrow \ \varphi < \Phi_2(\theta))$,
- $\varphi = \Phi_2(\theta)$ and $\varphi > \Phi_1(\theta)$ $(\Rightarrow \ \varphi > \Phi_3(\theta))$,
- $\varphi = \Phi_3(\theta)$ and $\varphi > \Phi_2(\theta)$ $(\Rightarrow \ \varphi < \Phi_1(\theta))$.

Let us have a look at these discontinuities when we are on the curves $\varphi = \Phi_i(\theta)$, figure 2:
$\forall i \in 1,2,3$,

$$\begin{aligned} \frac{\partial d_i}{\partial \theta} &= y_i \cos\theta\cos\varphi \\ \frac{\partial D_i}{\partial \theta} &= (y_i - y_j)\cos\theta\cos\varphi \\ \frac{\partial D_i}{\partial \theta}\Big|_{D_i=0} &= (y_i - y_j)\frac{\cos\theta}{\sqrt{1+\left(\frac{y_i-y_j}{x_i-x_j}\right)^2\sin^2\theta}} \end{aligned} \quad (4)$$

$(4) \Rightarrow \frac{\partial D_i}{\partial \theta}\Big|_{D_i=0} \neq 0$ for $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ since $y_i \neq y_j$ and $\cos\theta$ is non-null.

With the same approach, $\forall i \in 1,2,3$,

$$\begin{aligned} \frac{\partial d_i}{\partial \varphi} &= -x_i\cos\varphi - y_i\sin\theta\sin\varphi \\ \frac{\partial D_i}{\partial \varphi} &= (x_j - x_i)\cos\varphi + (y_j - y_i)\sin\theta\sin\varphi \\ \frac{\partial D_i}{\partial \varphi}\Big|_{D_i=0} &= \frac{x_j - x_i}{\cos\varphi} \end{aligned} \quad (5)$$

$(5) \Rightarrow \frac{\partial D_i}{\partial \theta}\Big|_{D_i=0} \neq 0$ for $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ since $x_i \neq x_j$.

C. 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. Consequences of the strict convexity of one body are given by the following theorems:

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.

Proof: If both objects are not convex, then when their non convex parts are parallel, there is an infinite number of witness points pairs.
If one object is strictly convex, on the contrary, supposing the existence of two distinct witness pairs leads to a contradiction because another pair can be found in between, for which the distance is strictly smaller. ■

Theorem 2.2: 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: See Appendix 1. ■

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.

Proof: Let's derive $\delta(q) = \|f(q, u_{\min}(q))\|$ with respect to q:

$$\begin{aligned}\frac{\partial \delta}{\partial q}(q) &= \left(\frac{\partial f}{\partial q} + \frac{\partial u_{\min}}{\partial q} \cdot \frac{\partial f}{\partial u}\right)^T \frac{f}{\|f\|} \\ &= \left(\frac{\partial f}{\partial q}(q, u_{\min}(q))\right)^T \frac{f(q, u_{\min}(q))}{\|f(q, u_{\min}(q))\|} \quad (6)\end{aligned}$$

since $\left(\frac{\partial f}{\partial u}(q, u_{\min}(q))\right)^T f(q, u_{\min}(q)) = 0$
The result is obtained by composition, f being C^∞ with respect to q and of the C^k with respect to u, where k is the minimum continuity index of the bodies' surfaces. ■

So far, we did not require anything of the objects but to be continuous and convex (strictly for one). However, 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.

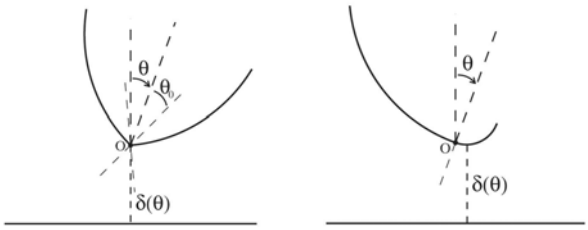
Proof: Let u_0 be the coordinates of the witness points at q_0.
We have $\left(\frac{\partial f}{\partial u}(q_0, u_0)\right)^T f(q_0, u_0) = 0$ (optimality condition) which can be rewritten $\frac{\partial f^2}{\partial u}(q_0, u_0) = 0$.
For a given q_0, f^2 is the square distance between two points 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. ■

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.

Proof: We use eq. (6) once again. ■

Note that having C^1 surfaces does not improve the continuity of the distance compared to C^0 surfaces as shown by the following example (see fig. 3).

Let's consider, in a 2D space, 2 arcs of circle of same radius R, joining at the origin O of the world frame and being part of a strictly convex object O_1, so that O_1 is only C^0. θ denotes the angle between the vertical axis and the symmetry axis, θ_0 the non oriented angle between the diameter of one circle going through O and the symmetry axis. We consider



(a) C^0 case (b) C^1 case

Fig. 3. Distance between a plane and a C^0 or C^1 object.

the distance $\delta(\theta)$ from this object to a line $z = -z_0$.

$$\delta(\theta) = \begin{cases} R(\cos(\theta_0 - \theta) - 1) + z_0 & \text{if } \theta < -\theta_0, \\ z_0 & \text{if } -\theta_0 \leq \theta \leq \theta_0, \\ R(\cos(\theta - \theta_0) - 1) + z_0 & \text{if } \theta > \theta_0 \end{cases} \quad (7)$$

This function is C^1, but not C^2:

$$\delta''(\theta) = \begin{cases} -R\cos(\theta_0 - \theta) & \text{if } \theta < -\theta_0, \\ 0 & \text{if } -\theta_0 \leq \theta \leq \theta_0, \\ -R\cos(\theta - \theta_0) & \text{if } \theta > \theta_0 \end{cases} \quad (8)$$

Now we consider two arcs of circle of radii r and R $(R > r)$, whose centers are aligned with the origin O. Surface is C^1, and

$$\delta(\theta) = \begin{cases} R(\cos(\theta) - 1) + z_0 & \text{if } \theta < 0, \\ r(\cos(\theta) - 1) + z_0 & \text{if } \theta \geq 0 \end{cases} \quad (9)$$

And we still have a discontinuity of the second derivative, due to the change of radius:

$$\delta''(\theta) = \begin{cases} -R\cos(\theta) & \text{if } \theta < 0, \\ -r\cos(\theta) & \text{if } \theta \geq 0 \end{cases} \quad (10)$$

However the discontinuity is less important in this case: $R - r$ whereas it was R in the first case.

To sum up the continuity properties of the minimum distance between two convex bodies:

- the distance is always C^0 (even with no convexity assumptions),
- it is piecewise C^1 with simple convexity. Discontinuities of the gradient arise when faces or edges are parallel (non-strict convexity),
- strict convexity of a body ensures C^1 property of the distance. Discontinuities of higher order arise whenever there are surface discontinuities,
- additional C^k property of both surfaces yields C^k smoothness of the distance.

These properties are also true locally. In particular, if witness points move on the interior of C^∞ surfaces, one at least being strictly convex, the distance is C^∞. This is for example the case when considering the distance between 2 spherical parts of the bodies.

Fig. 4. Degenerate case where there is an infinity of witness pairs (some pairs are depicted in gray).

D. 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 anymore. This implies that in some configurations there are several (up to a non-countable infinity, see fig. 4) pairs of witness points. Jumps between witness pairs are thus inevitable so that there are gradient discontinuities. 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). As said before we can have several possible pairs and consequently jumps of these pairs. This is however not the case if the penetration depth is less than the minimal curvature of the penetrating parts of the objects. Under this assumption of "slight" penetration the previous results hold.
If this assumption is violated, 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, so they should not be a problem in most applications where penetration can occur (like optimization beginning with an unfeasible starting point).

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 own type of feature (faces with more than 3 vertices are cut into triangles for this discussion):

- 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. Each part of sphere or torus is called *feature*.
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.

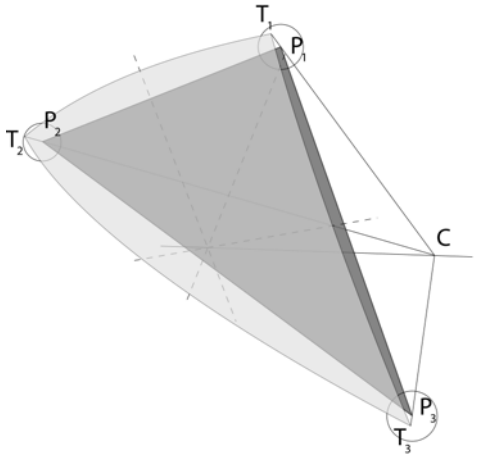


Fig. 5. Sphere construction.

A. Big spheres construction

We consider a triangular face of the polyhedron (fig. 5). 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 must be 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.
Let's denote $u = P_1P_2$, $v = P_1P_3$, $c = P_1C$ and $w = u \times v$. w is collinear to the outer normal and points in the same direction.
C needs to be equidistant to the points P_i and is thus on the median planes of the edges. We then solve the following system to find the coordinates of C in the frame (P_1, u, v, w):

$$\begin{cases} u.c &= u^2/2 & \text{(median plan of } P_1P_2) \\ v.c &= v^2/2 & \text{(median plan of } P_1P_3) \\ c^2 &= (R-r)^2 & \\ c.w &< 0 & \text{(inner solution)} \end{cases} \quad (11)$$

We write $c = \alpha u + \beta v + \gamma w$ to use the fact that w is orthogonal to both u and w.
The two first equations become a linear system whose solution is

$$\begin{cases} \alpha &= \dfrac{u^2v^2 - u.v \cdot v^2}{2(u \times v)^2} \\ \beta &= \dfrac{u^2v^2 - u.v \cdot u^2}{2(u \times v)^2} \end{cases}$$

These are the coordinates of the circumcenter of the triangle. Replacing α and β in the third equation leads to an equation of degree 2 in γ, which is simply a way of writing the

Pythagorean theorem in the triangle made by P_1, C and the circumcenter. Only one of its 2 solutions complies with the fourth equation: $\gamma = -\sqrt{\frac{(R-r)^2-\alpha^2u^2-\beta^2v^2-2\alpha\beta u.v}{(u\times v)^2}}$
Of the sphere centered on this point C, we only keep the part inside the cone defined by vectors CP_i and whose apex is C.

B. Toruses construction

Roughly speaking, we obtain the torus above one edge by rotating the sphere of a neighbouring face around this edge and keeping the resulting inner volume. The construction is based on the following result:

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. ■

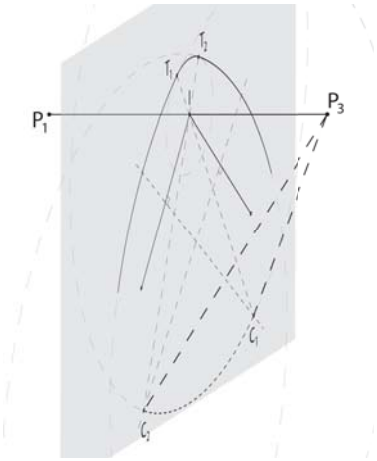


Fig. 6. Torus construction.

The centers C_1 and C_2 of the two spheres corresponding to the two neighbouring faces of the edge we consider on fig. 6 are thus on a same circle centered on 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.
By making it revolve around the edge until it coincides with the circle $\mathcal{C}_2$ of same radius centered in C_2 and in the plane defined by C_2 and the edge, we obtain the part of torus we need. It should be noticed 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 7.

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.

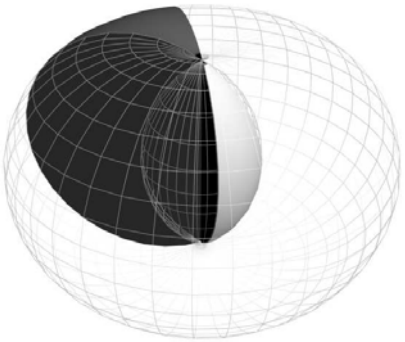


Fig. 7. Torus part. The black and white parts are the portion of torus we consider, the white surface is the part used in the STP-BV

C. Properties

Theorem 3.2: In the STP-BV, the toruses are tangent to the small spheres.

Proof: Each torus is obtained by the revolution of a circle around an edge that is also a revolution axis of the small sphere, and the circle is tangent to the small sphere. ■

Theorem 3.3: (Continuity and convexity properties of STP-BV) The STP-BV is C^1 and strictly convex.

Proof: All parts of the STP-BV are C^1 and strictly convex, and are tangent wherever they connect. ■

STP-BVs are even piecewise C^∞ since toruses and spheres are C^∞ surfaces.

Theorem 3.4: (Maximal margin) 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 $R-\sqrt{(R-r)^2-\frac{a^2}{12}}$

Proof: The maximal distance between a vertex of the convex hull and the STP-BV is r.
Since each edge is the revolution axis of its associated torus, maximal distance is achieved in its median plane, and is equal to $R-\sqrt{(R-r)^2-\frac{l^2}{4}}$, l being the length of the edge.
For a face, two cases arise regarding the position of the circumcenter H:

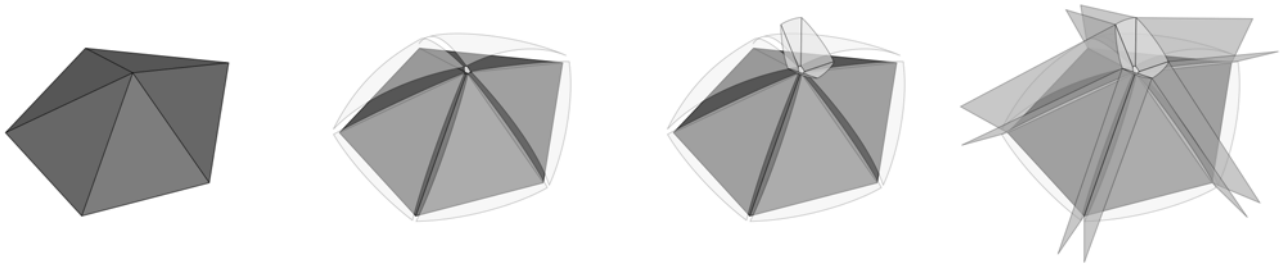
- H is outside of the face and then maximal distance is achieved on the longest edge of this face,
- H is inside the face. In this case, for a fixed longest length of the edges, the maximal distance is achieved when the face is equilateral and is $R-\sqrt{(R-r)^2-\frac{a^2}{12}}$.

■

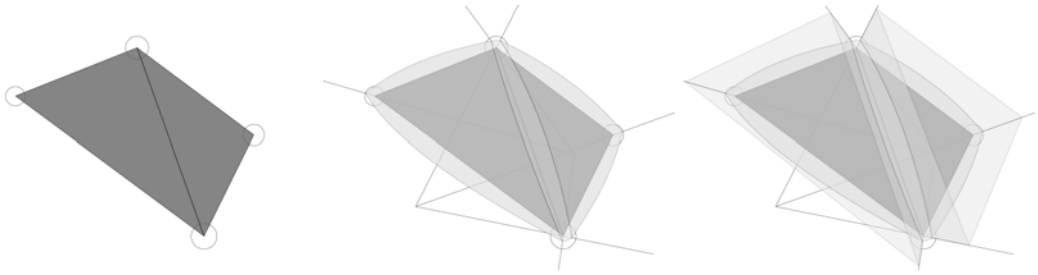
If R is notably bigger than r and a, the expression can be accurately approximated (Taylor expansion) simply by r.
For example, with the values we typically use in our applications ($a = 10$cm, $r = 1$cm and $R = 10$m), the maximal margin is 1.00417cm.

D. Voronoi region

An edge is the revolution axis of both its associated torus and the spheres of its vertices. Therefore finding the voronoi



(a) voronoi regions around a vertex. From left to right: vertex with its adjacent faces, small and big sphere ot the STP-BV, voronoi region of the small sphere, all the voronoi limits around the vertex.



(b) voronoi regions around an edge. From left to right: edge with its adjacent faces and the small spheres attached to the vertices, STP-BV, voronoi regions of the faces.

Fig. 8. Voronoi regions.

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 in the very same way as in figure 3(b). 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 and according to the previous remark, this plane is 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 here is the line defined by the centers of both spheres. This line is also part of all the limits between neighbouring voronoi regions. Illustrations are given with figures 8.

E. Advantages

As stated in theorem 2.5, there is no advantage in having C^1 surfaces instead of C^0 regarding the distance continuity. Thus we could have used a bounding volume made only of parts of spheres, one above each face of the polyhedron. Distance would have been C^1 nonetheless, since the polyhedron would have been round off. However, STP-BV has the following advantages on such a bounding volume (that we will call hereafter SP-BV):

- closer to the original object for a same minimal safety margin: in case of a sharp edge, sphere of SP-BV would intersect far from the edge since maximal margin for SP-BV is $2R$ (achieved when the polyhedron tends to a point),
- same complexity of distance computation, since in both cases we need point-point, point-circle and circle-circle computation,
- as for voronoi regions, the relationship between STP-BV and polyhedrons is simpler that between SP-BV and polyhedrons,
- normal vector is well-defined everywhere along the surface, which is important for gradient computation, especially when the normal cannot be derived from the witness points, as it is the case when contact arises (see section V),
- the minimal curvature radius of STP-BV is r whereas it is 0 for SP-BV: STP-BV handle thus better the slight penetration case (see II-D).

Additionally, the jumps of the second derivatives is slightly smaller for STP-BV than for SP-BV since it is related to the ratio between the radii of small and big spheres.

IV. Computing Proximity Distances

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 data.
Prior to the calculations, STP-BVs have to be built, as well as some other data regarding the voronoi regions of the volumes. These calculations are made off-line for each object.
In this section we will name *polyhedral feature* a feature of the object (i.e. a vertex, an edge or a face) and *smooth feature* a feature of its STP-BV (i.e. a part of small sphere, big sphere or torus) to avoid any confusion.

A. Bounding volume construction

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**
    $(e, v) \leftarrow$ *first*(edgeStack)
    $(e_1, e_2) \leftarrow$ *newEdges*(e,v)
    push(output, *face*(e,v))
    if contains*(edgeStack,* e_1*)* **then**
      delete(edgeStack, e_1)
    else
      insert(edgeStack, e_1, *angleMin*(e_1))
    end
    if contains*(edgeStack,* e_2*)* **then**
      delete(edgeStack, e_2)
    else
      insert(edgeStack, e_2, *angleMin*(e_2))
    end
  end
  return output
end

Algorithm 1: Bounding volume construction.

The construction of the bounding volume is depicted in algorithm 1. It does not rely on the faces of a polyhedron, but only on its vertices, i.e. it works on a cloud of points from which it rebuilds faces.
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.

Algorithm: Bounding volume construction intermediate functions: Pseudo-code

init(edgeStack, faceList);
begin
  for *each* (v_1,v_2,v_3) $\in vertices^3$ *with* $v_i \neq v_j$ **do**
    $s \leftarrow$ *sphere*(v_1,v_2,v_3);
    if allPointsInSphere*(s)* **then**
      for $i \in \{1, 2, 3\}$ **do**
        $e_i \leftarrow$ *edge*(v_i, $v_{(i+1)[3]}$,$v_{(i+2)[3]}$);
        insert(edgeStack, e_i, *angleMin*(e_i));
      end
      push(faceList, *face*(e,v));
      return SUCCESS;
    end
    return FAIL;
  end
end

angleMin(e);
begin
  (v_1,v_2,v_3) $\leftarrow e$;
  $s \leftarrow$ *sphere*(v_1,v_2,v_3);
  for *each* (v) $\in vertices - \{v_1, v_2, v_3\}$ **do**
    $s' \leftarrow$ *sphere*(e,v);
    if allPointsInSphere*(*s'*)* **then**
      a = *angle*(s, s', e);
      return (a,v);
    end
  end
  return FAIL;
end

Algorithm 2: Bounding volume main sub-functions.

Finding the first face is made by the *init* function (algorithm 2). Since we can only rotate around a single edge at a time, it is needed to have the list of edge around which haven't yet rotated. Although we could chose any edge from this list to go for the next step, we select the edge around which the smallest rotation will be required (the explanation of this choice is given later on in this section). Therefore, the edge must be stored with some additional data, so that the angle can be computed. The edge structure thus contains three vertices: the two extremities of the edge and an additional one, corresponding to the third vertex of the face the edge already belongs to. We now know where the rotation is made from and in which direction.
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 according to the value of its associated angle. Updating this list is the main task of *buildVolume*, through the classical functions *first*, *insert*, *delete* and *contains*. *first* deletes the element after reading it.
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*, it means that 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, which is the angle made by the center of these spheres and the midpoint of the edge.

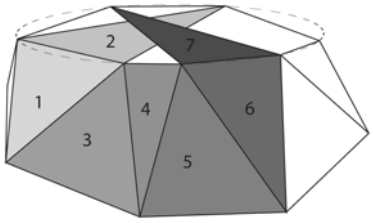


Fig. 9. Misbehavior of the building algorithm when facing a polygonal face whose points are on a same circle. The algorithm builds the faces in the order given by the numbers. When turning around the upper edge of face 6, the point chosen is not the same as the one chosen when building face 2 from face 1, resulting in overlapping triangles.

The reason for choosing the edges the way we do is to ensure robustness of the algorithm regarding numerical errors in the case of polygonal faces whose vertices are on a same circle. In this case all vertices should be reached at the same time when turning around an edge of this face and thus the vertex with the lowest index would be chosen. However this is sometimes not the case because of numerical rounding errors. Different vertices can thus be chosen when we arrive on the face by turning around different edges, resulting most of the times in overlapping triangles, which makes the algorithm fail (see figure 9). To avoid this, we force the algorithm to finish to cover a polygonal face it already began, by choosing to turn around the edge with the lowest needed rotation. We then also avoid to give a threshold that defines when points are coplanar, and to search for coplanar points each time there is a rotation around an edge.

Our algorithm computes a kind of convex hull for an object, as well as the big spheres corresponding to each face of this hull. From the spheres and the faces, the STP-BV is easily deduced along with its voronoi regions.
The obtained hull is not the convex hull, for some points of it may have been ignored because of the curvature of the spheres, as shown in figure 10 in a 2D example. However, when r tends to 0 and R becomes infinite, this hull tends to the convex hull of the cloud of points.

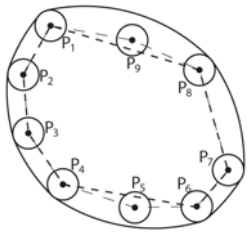


Fig. 10. Difference between the obtained hull and the convex hull in a 2D case. Because of the curvature of the circle, points P_5 and P_9 and their associated small spheres are strictly inside the STP-BV and thus not on the underlaying hull (thick small dashed line) whereas they belong to the convex hull (thin long dashed line).

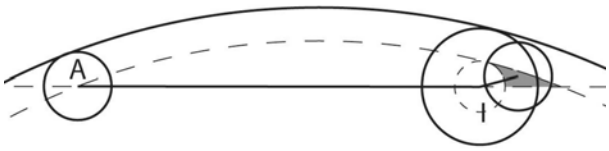


Fig. 11. A case where the obtained hull is not convex: for simplicity, we consider an isosceles triangle ABC and look at it in its symmetry plane. Circle centered in A is the small sphere of this vertex, plain circle centered in I is the torus of BC and plain big circle is the sphere corresponding to the face. Dashed circles correspond to the same spheres and torus whose radii have been decreased by r. Any point in the dark area may be accepted by the covering algorithm when turning around BC while it is above the previous face.

Remark: The obtained hull may even be non convex in very rare particular cases involving very thin triangles as exemplified in fig 11. This can be a problem since we want to use classical distance computation algorithms that usually work with convex polyhedrons. It can be solved in different ways, either by moving or removing the guilty vertex, or by cutting the hull in convex parts. However experiments with large variety of objects from real applications never exhibited such a case. We thus did not program anything to handle it, we simply check the convexity of the obtained hull.

B. Overall algorithm

Computing the distance for two (non necessarily convex) polyhedra O_1 and O_2 is done in two steps (fig 12).

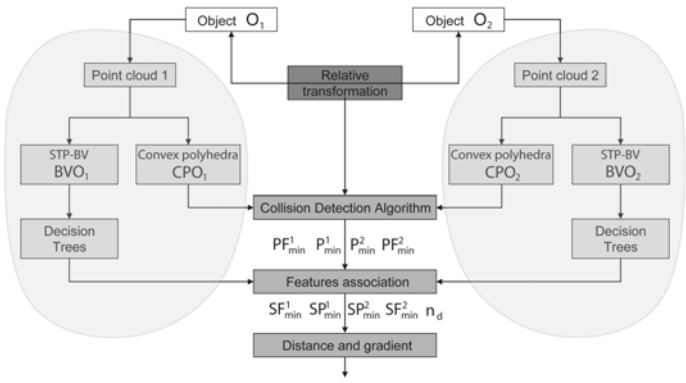


Fig. 12. Overall algorithm. The steps in the gray area are computed off-line.

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 implementation of this part has not been optimized simply because it occurs off-line. 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.

The computation of the distance and theses data requires to know how to find them for three kinds of pairs of smooth features: sphere-sphere, sphere-torus and torus-torus. The first case is trivial, the two other ones are detailed in the section VI.

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 as described in the following subsections,
- 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).

For example, if PF_1 is a vertex and SF^0_1 its associated small sphere, the position of $P^2_{\min}$ regarding the voronoi region of SF^0_1 decides whether SF_1 is SF^0_1 or one of the adjacent toruses or (big) spheres.

As shown in V-Clip introducing paper [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 $SP^1_{\min}$, $SP^2_{\min}$, SF_1 and SF_2. Tests with objects such as used in the examples of section VII 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. However, the witness points pairs found without correction are in most cases really close to the correct ones (error between points are less than $0.1mm$). We thus chose not to make corrections, for the sake of speed.

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 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. 13. 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. To speed up the process even more, a tree of tests is built during the off-line process in order to minimize the average number of tests needed to determine the region.

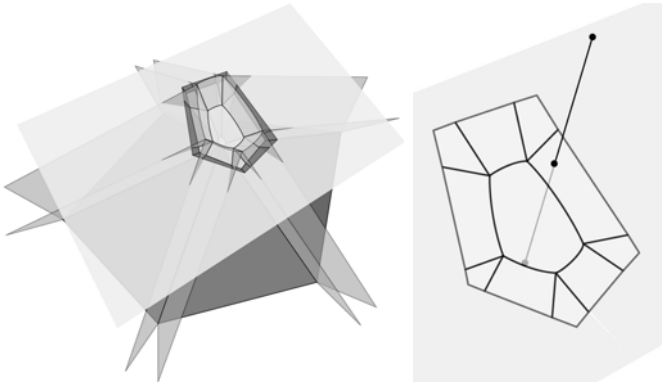


Fig. 13. Intersection of a plane with the voronoi regions related to a vertex. Smooth voronoi regions are those from fig 8, darker cone is the voronoi region of the vertex.

To sum up: if PF_i is a vertex, SF_i can be the associated small sphere or one of the smooth features associated to the edges and faces PF_i belongs to. $P^j_{\min}$ is projected in a plane where the corresponding smooth voronoi regions have already been projected. The projection of $P^j_{\min}$ determines SF_i.

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 intersect in various ways with the regions of toruses/edges, depending on the shapes of the faces. In all cases, the tests to be made are to know on which sides of the smooth voronoi limit planes the witness point is.

When a (triangle) face is non-obtuse, the smooth voronoi region of its associated big sphere contains its polyhedral voronoi region. The center of the sphere projects inside the face so that the angle between the face and the limits of the voronoi region of the sphere is greater than 90 degrees. There is thus no test to do: the smooth feature is this sphere.

With the same argument, if an edge is between two non-obtuse faces or is not the longest side of an obtuse face, then its polyhedral voronoi region contains the smooth region of

its torus. A witness point in the polyhedral region can then be either in the torus region or the regions of the two adjacent spheres. A test is there needed to determine on which side of the limit planes the point is.

If a face is obtuse, then its polyhedral voronoi region intersects with the smooth region of the torus associated to its longest edge. A single test is then needed to know wether the witness point is inside the smooth region of the associated big sphere or not.

If an edge is the longest one of an obtuse face then there is no test to perform to know whether the witness point is in the smooth voronoi region of the face's big sphere.

A problem arises if a face is really thin while the surrounding shape of the object is quite flat: it may happen that smooth voronoi regions intersect several polyhedral voronoi regions in addition to the previously cited ones. However, when the radius of the big spheres R is large compared with the object size, such an intersection is unlikely to occur or will occur very far from the object, at a distance where the error would not impact the output distance or its gradient. This is why we kept the solution:

- if PF_i is a non-obtuse face then SF_i is the associated big sphere,
- if PF_i is obtuse, a test has to be made to know whether SF_i is the big sphere or the torus associated to the longest of PF_i,
- if PF_i is an edge and not the longest edge of an obtuse face, the position of $P^j_{\min}$ regarding two planes has to be known to determine whether SF_i is the torus or one of the sphere associated to the adjacent faces,
- if PF_i is the longest edge of a single face, a test has to be made to know whether SF_i is the torus or the sphere associated to the other face,
- if PF_i is the longest edge of both its adjacent faces, then SF_i is its associated torus.

V. 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:

$$\frac{\partial \delta}{\partial q}(q) = n_d^T \left(\frac{\partial SP^1_{\min \in BVO_1}}{\partial q}(q) - \frac{\partial SP^2_{\min \in BVO_2}}{\partial q}(q) \right) \quad (12)$$

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.

We only have a small but important difference with [12]: our normal unit vector n_d is not derived from the position of the smooth witness points, but directly computed from the smooth features. Indeed, in the case of a contact between the objects, the witness points coincide and thus cannot define a vector. Since for example in an optimization process distance constraints may prevent from reaching the criterion minimum, such a contact case is not unlikely to occur, and actually did when we generated postures as presented in VII.

In the penetration case however, when there are multiple witness point pairs, it is not possible to define this vector, but the gradient is undefined anyway.

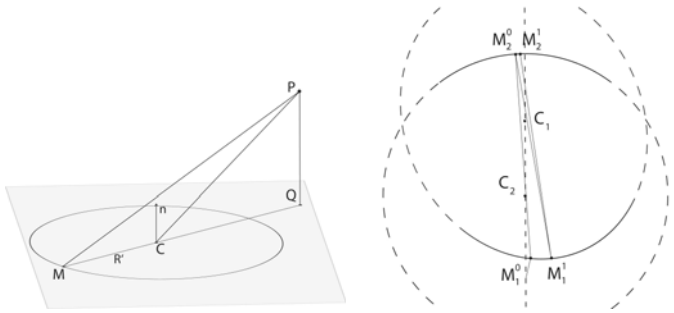
For a point P of fixed coordinates (x, y, z) in the local frame of an object O at the configuration(q), the gradient has the following expression: $\frac{\partial P}{\partial q}(q) = xJ_1(q) + yJ_2q(q) + zJ_3(q) + J_4$ obtained by deriving $P(q) = R(q)(x, y, z)^T + T(q) = xC_1(q) + yC_2(q) + zC_3(q) + T(q)$ 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*.

VI. Implementation

For the implementation of our algorithm, 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.



(a) point-circle distance (b) circle-circle distance

Fig. 14. Basic distance computation.

However these algorithms cannot be used directly: 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 to do. With the notation of figure 14(a), the distance is $\sqrt{R'(R' + 2\,\|\overline{CQ}\|) + CP^2}$.

For the circle-circle distance, we could adapt the algorithm in [14] to find a maximum instead of a minimum, and stop at the first maximum. This is possible because, since we use only part of circle, we have a single maximum. It appeared

however that in our case, the following iterative algorithm was more efficient (fig. 14(b)): we first determine a point on the circle associated to the feature SF_1 by taking the intersection of this circle with the projection onto its supporting plane of the line passing through the centers of the circles, C_1 and C_2. We chose the intersection M_1^0 so that $C_1C_2.C_1M_1^0 > 0$. From this point, we compute M_2^0, the farthest point on the second circle. Then M_1^1 from M_2^0 and so on until convergence is achieved.

B. Computation time

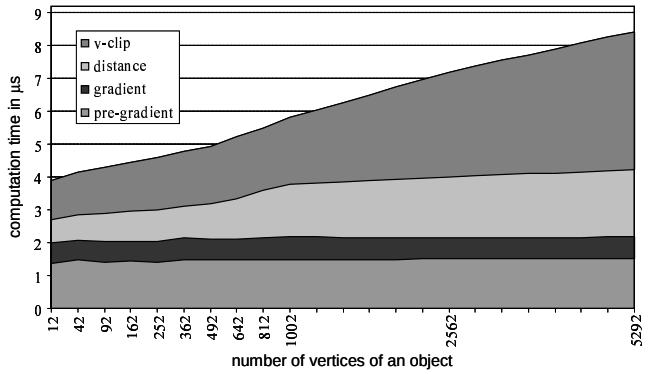


Fig. 15. Computation time for small relative movements

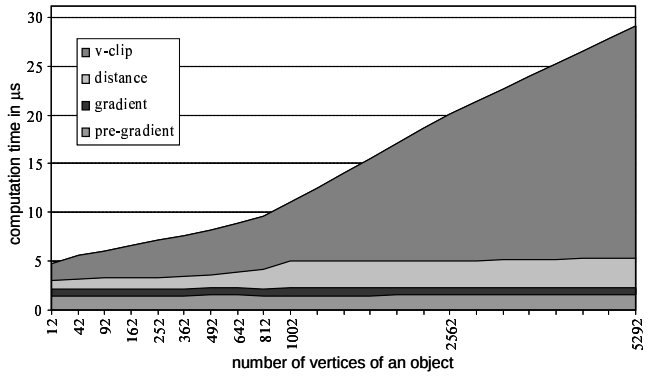


Fig. 16. Computation time for large relative movements

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 (directly related to the number of faces : $f = 2(v-2)$ with f number of faces and v number of 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. In the results presented in 15, the rotation angles are about 2 degrees and the average distance change is 2mm. For the graph in fig. 16, angles are about 20 degrees and the average distance change is 2cm. Angle increments are taken so that the same configuration never appears twice.

In both cases, the results are given for one average call. As expected, only V-Clip is sensitive to the amount of variations between two consecutive configurations, and the continuous gradient layer is almost in constant time with respect to the number of vertices: pre-gradient and gradient computations are done in constant time of 2 microseconds (1.4 and 0.6 respectively), the distance computation which mainly computes the feature associations slightly increases (almost like a square root) with the number of vertices. This is most certainly due to the larger amount of data attached to more complex objects but this increase is small compared with V-Clip's . We have no explanation so far for the sudden increase between 812 and 1002 vertices.

Computations were performed on a 3.4GHz Pentium Xeon with 2GB of RAM.

VII. Application: Free-collision Humanoid Posture Generation

Now that the method is explained, and packaged into a C++ code, we will demonstrate it in actual robotic context as announced in the introduction section. 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.

Roughly speaking, 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:

$$f(q) = \sum_{i=1}^{n_j} \left(q_i - \frac{q_i^{\min} + q_i^{\max}}{4} \right)^2 \tag{13}$$

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 17. For human likeness, some more advanced but also more time-costly criteria have already been suggested, such as those mentioned in [15]. The problem of collision avoidance in posture generation has also been addressed in [16] and in [17]. Authors in [16] generate postures from an optimization formulation; spheres have been used as a coverage to define collision avoidance constraints. The problem of their method is in the number of spheres needed to cover the virtual avatar while keeping a good precision of the shape, along with the combinatorial complexity of the sphere pairs to be considered as constraints. Authors in [17] generate postures from inverse kinematics and prioritized tasks, but this method is not smooth with all observers' primitives, moreover observers need to be specified by the user and are context specific.

Figure 17 shows both the geometrical model of 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

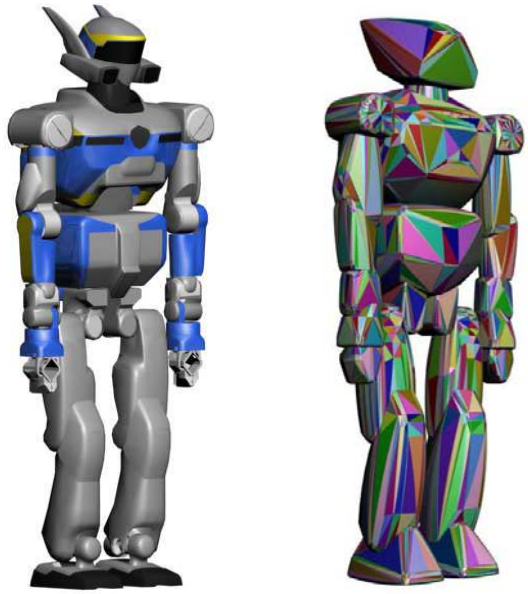


Fig. 17. HRP-2 robot and its STP-BV

to have few difference between safe margin and maximal margin).

Hereafter are snapshots of the optimized posture obtained in various scenarios, which were extracted from the usual working day of a HRP-2 robot. In all cases, the humanoid is asked to reach defined goals described here as 3D points (and an eventual orientation). We simply ask for matching specific points of the humanoid robot (e.g. a given point on the right gripper, another one on the left foot or another one on the left gripper...) with corresponding target 3D points in the free space. These matches are described as constraints to be satisfied in the optimization problem.

Concerning auto-collision, robot pairs that need to be selected for checking have been studied in [18] and recently in [19]. 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. 18. Collisions with the environment 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.

In this scenario, and much more in the third one, the robot is quite stretched and close to many of its joint limits because of collisions (with the environment in this case, mainly with itself in the at-work scenario). 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 and in the third scenario.

Enter-car scenario: a posture is tested where the robot has to place a foot on a car floor in front of the driver seat. Placement constraints here are to have the left foot on the floor outside the car and the right one inside. The robot must be stable with only the left foot contact. Collisions with the environment are checked for the driver seat, the door, its window, the dashboard, the steering wheel and parts of the bodywork. 28 pairs are checked. The robot is initially at its 0 posture and placed in the middle of the car (but is not colliding with parts that are checked).

Leftmost picture of fig 19 shows the output of the posture generator when collisions are not taken into account. Collisions between the left hand and the door as well as between the right knee and the dashboard can be seen. The next three pictures are different views of the collision-free posture obtained. Computation for this posture takes 79 milliseconds (16 without collision checking), over 23 iterations; the distance function is called 2,898 times and the gradient 2,882 times: although the space for the right leg is really narrow, the chest and the other limbs have a quite wide feasible space.

At-work scenario: HRP-2 is on a ladder in a warehouse and tries to reach a crate. For this scenario, both feet are placed on the same step, right hand is asked to grasp the banister and the left hand must reach a point far in the front-left side. Collisions are checked with the bodies and all the relevant ladder parts (not with the stairs behind the robot for example) so that there are 15 pairs checked. Initial posture is in front of the ladder, angles being at 0. This scenario was designed to exhibit some self-collisions, as can be seen on the second picture of fig. 20: there are collisions between the right arm and both waist and chest. Additional collisions occur with the environment: lower leg and step, thigh and banister. These collisions are avoided in the final posture shown in the leftmost and two rightmost pictures. Computation time in this case is 3.328 seconds, the distance function is called 54,993 times and the gradient 33,000 times. FSQP needs 265 iterations.

Fig. 18. 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).

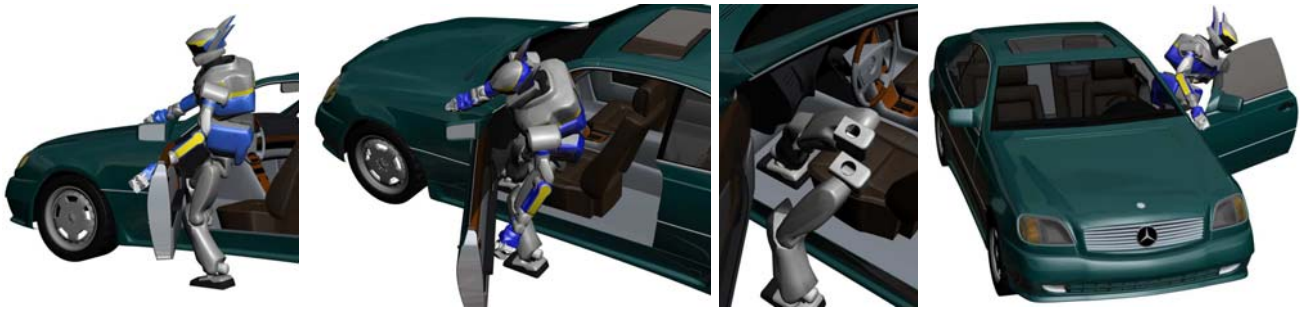


Fig. 19. Enter-car scenario. From left to right: without collision avoidance constraints, successful free-collision posture from three different view points.

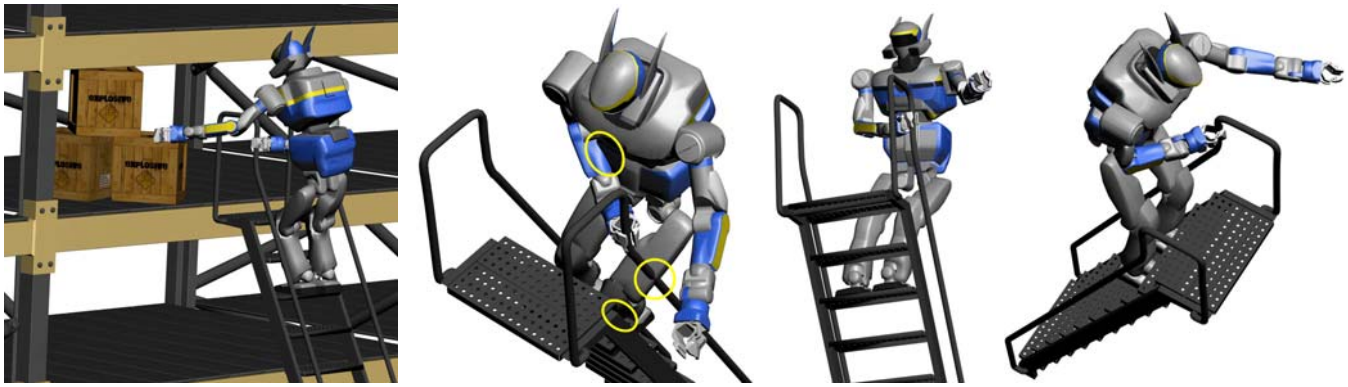


Fig. 20. At-work scenario. From left to right: general view of the result and the context, posture obtained without collisions avoidance constraints, successful free-collision posture from two different view points

VIII. Discussion

If we increase the radius of the big spheres, the STP-BVs flatten. Consequently, the witness points travel faster on the objects for a same relative movement, so that the gradient's variations are larger: the bigger the radius is, the closer we are to the non-strictly convex case, with its gradient discontinuities. This has been noticed when using $R = 100m$: the posture generator converged more slowly in this case. The difference is noticeable in the *At-work* scenario where the generation needs 5,016 seconds for 393 iterations (with 88,505 calls to the distance routine and 49,000 to the gradient), and huge with the *fridge* scenario (3,078 seconds, 282 iterations, 68,931 distance calls and 37,935 gradient calls).

As stated in theorem 2.3, gradient continuity is achieved as soon as one object is strictly convex. There is then no particular need to have both objects of a pair covered with STP-BV, and we could compute C^1 distances between one STP-BV and one polyhedral object. However computation time in this case is not expected to decrease because the complexity is roughly the same as for two STP-BV: features have to be found for the two objects from the output polyhedral features of the classical detection algorithm (there may be a change of feature for the polyhedral object), and the most costly elementary distance computation for two STP-BV, namely the circle-circle distance computation, would be replaced by a line-circle distance computation the cost of which is even greater (see [14]).

Considering the on-line computation time, it is thus not

interesting to develop an algorithm for STP-BV/Polyhedron distance computation. Still, such an algorithm would have an advantage regarding the pre-computations: if each object needs a STP-BV, every time a new object is inserted in a scenario or an object is changed, some off-line computations have to be done; if a convex polyhedron is kept, only voronoi regions have to be determined (which is already done by V-Clip for example) as well as their graph. It would be more user-friendly to have to compute STP-BV for objects that seldom change, while ensuring that every checked pair has at least one STP-BV. In the case of our application, this simply means that the robot has to wear STP-BVs while the objects of the environment remain as polyhedrons.

This issue is very important in robotics because it opens doors for hardware porting and exploitation in designing robust low-level control implementations. Continuity of the proximity gradient makes it possible to be also used for tracking tasks other than avoiding self-collisions and collisions. For example, devising reactive controllers for exploring motions at a predefined distance along a given object/obstacle of the environment can possibly benefit from proximity gradients continuity. This issue is being actively investigated by our team for the humanoid HRP-2. Note that the intrinsic problems that may occur is the penetration case, see subsection II-D, will not be practical control problems.

It must also be pointed out that the computation cost for the pre-gradient matrices is quite heavy, as shown in the figures of section VI. However, it is distributed between the gradient computations of most of the constraints - not only the collision constraints. These constraints are therefore computed a bit faster in the case of the posture generator case.

IX. Conclusion

A new method for computing proximity distances with continuous gradient has been presented. 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 the following points:

- the guaranteed continuity properties of the proximity distance allows its inclusion as part of the constraints in the up-coming implementation of the multi-sensory task sequencing for sensor-based control of the humanoid HRP-2 [20];
- the packaged software will be also part of the optimization software which is being developed for robotic trajectories generation [4];
- research on a methodology which allows automatic and minimum necessary body pair assignments for auto-collision checking in a poly-articulated structure;
- V-Clip is used as an intermediary step for computing the proximity distance between a pair of bodies. We consider getting rid of this step by designing a new version of the proximity distance algorithm, certainly on the same basis as V-Clip, which would define Voronoi regions and compute distance using directly the STP-BV, and possibly adapt it to distance between STP-BV and polyhedrons. This will definitely eliminate the problem of fig. 11 and the approximations made for the smooth features association;
- in the same way as for auto-collision, we aim at reducing the number of constraints to be used between the robot and objects composing its surrounding environment. Space partitioning techniques and bounding volume hierarchies have proved to be efficient and we are likely to use them.

Appendix I
Proof of theorem 2.2

Theorem 2.2: 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.

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.

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 $P_{0,\Delta}^2$ the new position of the point P_0^2 and $P_{\min}^1$ and $P_{\min}^2$ the new witness points so that $p_{\min}(q_0 + \Delta q) = (P_{\min}^1, P_{\min}^2)^T$.

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 $P_{\min}^1$ and $P_{\min}^2$. $\mathcal{P}_2^2$ (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 $P_{\min}^1$ (resp. $P_{\min}^2$). $\mathcal{S}^2$ is the sphere tangent to O_2 in $P_{\min}^2$ with a radius R. Its center C^2 is aligned with $P_{\min}^1$ and $P_{\min}^2$. 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 $\mathcal{C}_{\Delta q}^1$ (resp. $\mathcal{C}_{\Delta q}^2$) as the cylinder of axis $\left(P_{\min}^1 P_{\min}^2\right)$ 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 $P_{\min}^1$ and P_0^1, obtained for the extreme case where $P_{0,\Delta}^2$ is on the boundary of $\mathcal{C}_{\Delta q}^2$.

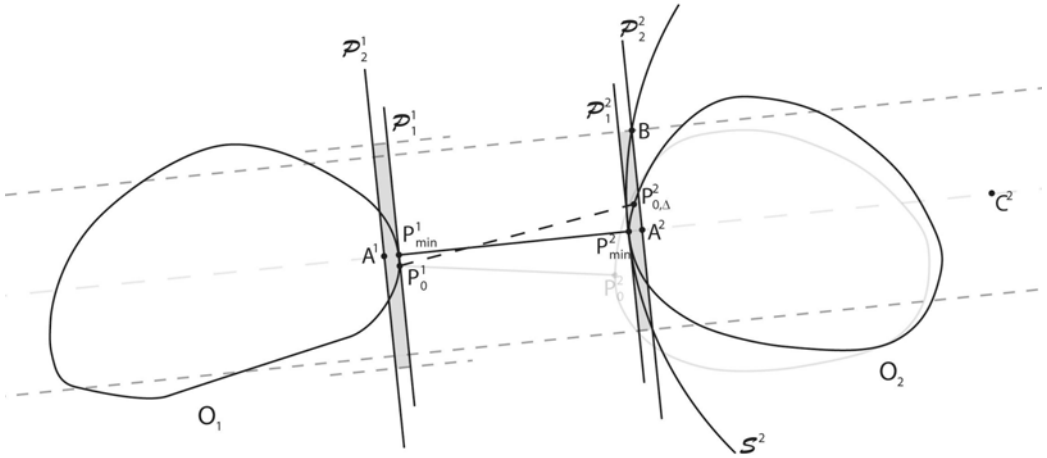


Fig. 21. Demonstration of the continuity of $P_{\min}$.

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 $\left(P^1_0, P^2_{0,\Delta q}\right)$ 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 $\left\{\left(P^1_0, P^2_{0,\Delta q=0}\right)\right\} = \left\{\left(P^1_0, P^2_0\right)\right\}$.
We then have $p_{\min}(q_0 + \Delta q) = \left(P^1_{\min}, P^2_{\min}\right)^T$ tends to $p_{\min}(q_0) = \left(P^1_0, P^2_0\right)$ which demonstrates the continuity in q_0. ■

ACKNOWLEDGMENT

The authors would like to thank...

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] K. Harada, K. Hauser, T. Bretl, and J.-C. Latombe, "Natural motion generation for humanoid robots," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, 2006, pp. 833–839.
[16] K. Abdel-Malek, J. Yang, T. Marler, S. Beck, A. Mathai, X. Zhou, A. Patrick, and J. Arora, "Towards a new generation of virtual humans," *International Journal of Human Factors Modelling and Simulation*, vol. 1, no. 1, pp. 2–39, 2006.
[17] M. Peinado, R. Boulic, B. Le Callennec, and D. Méziat, "Progressive cartesian inequality constraints for the inverse kinematic control of articulated chains," in *EuroGraphics*, 2005.
[18] 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.
[19] 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.
[20] N. Mansard and F. Chaumette, "Task sequencing for sensor-based control," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, February 2007.

PLACE PHOTO HERE

Adrien Escande is PhD student at the university of Évry. He graduated from École Nationale Supérieure des Mines de Paris (2005) with major in Robotics, where he participated to the French robotic cup. His main topic of research is contact planning for humanoid robots and virtual human avatars. He worked in the European project CAMELLIA and is currently partly involved in ImmerSence FP6 project. He joined the AIST/CNRS French-Japanese Robotics Laboratory (JRL) in 2006.

PLACE PHOTO HERE

Sylvain Miossec Sylvain Miossec is a research associate at the Centre National de la Recherche Scientiqgue (CNRS) working at the AIST/CNRS Joint Japanese-French Robotics Laboratory (JRL). He obtained his Master and Ph.D. from École Centrale de Nantes, France respectively in 2001 and 2004. He then obtained a JSPS fellowship for a two-years post-doctorate at the AIST/CNRS Joint Japanese-French Robotics Laboratory (JRL), until 2006. His research interests includes biped robots, humanoid robots, optimal motion, walking control and stability, multi-body simulation.

PLACE PHOTO HERE

Abderrahmane Kheddar is currently professor in computer science and control, and is the head of the virtual reality and haptics group of the University of Évry. He received a DEA (Master of science by research) and the Ph.D. degree in robotics, both from the University Paris 6, France. He was several months a visiting researcher at the formal Mechanical Engineering Laboratory (Bio-Robotics Division) in Japan. Since september 2003 he took a secondment position as a Directeur de Recherche at the Centre National de la Recherche Scientifique (CNRS), France. Since then he is the Codirector of the AIST/CNRS joint Japanese-French Robotics Laboratory (JRL) in Japan. His research interests include haptics (sensing, display and computer haptics), humanoid robotics, sensory and physically-based simulation, telerobotics and electro-active polymers. He is a founding member of IEEE RAS chapter on haptics. He coordinated the CNRS national specific actions, the one dealing with collision detection and the other on haptics. He is the coordinator of the FP6 European project ROBOT@CWE, and is the CNRS representative of the FP6 European network of Excellence in Virtual Reality INTUITION, the European coordination action in Presence PEACH, of the PF6 integrated project ImmerSence, the FP5 project TOUCH-HapSys.