# Strictly Convex Hull for Computing Proximity Distances with Continuous Gradients

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

***Abstract*— We propose a new bounding volume that achieves a tunable strict convexity of a given convex hull. This geometric operator is named STP-BV, which is the acronym for bounding volume made of patches of spheres and tori. The strict convexity of STP-BV guarantees a unique pair of witness points and at least $\mathcal{C}^1$ continuity of the distance function resulting from a proximity query with another convex shape. Subsequently, the gradient of the distance function is continuous. This is useful for integrating distance as a constraint in robotic motion planners or controllers using smooth optimization techniques. For the sake of completeness, we compare performance in smooth and non-smooth optimization with examples of growing complexity when involving distance queries between pairs of convex shapes.**

***Index Terms*— Continuous gradients of proximity distances, strictly convex hulls, sphere-torus patches, bounding volume, smooth and non-smooth optimization.**

## I. Introduction

THE DISTANCE FUNCTION and its use in collision-free robotic planning is remarkably addressed by the seminal work of Gilbert and Johnson [1] which provides a thorough analysis of its mathematical properties. In [1], motion planning is formulated as an optimization problem in which collision avoidance is defined as inequality constraints. Indeed, each collision avoidance constraint is written in terms of the Euclidian distance between the robot and a given obstacle geometry: this distance must always be above a certain positive threshold. A particularly important theorem in [1] –that we rediscovered and proved differently in [2]– is that the distance function for a pair of objects is not continuously differentiable unless one of the objects is convex and the other is strictly convex. As we illustrate later, non-differentiability of the distance function can be a problem when used in smooth optimization (see theoretical explanations in [3]): the optimization solver might not converge in some configurations. Then, why not use non-smooth optimization solvers? This is an option. However, with respect to smooth optimization routines, non-smooth ones are less easily available, less complete, and slower. Furthermore, it is worth adding distance constraints in existing schemes that already rely on smooth optimization routines (robotic models are generally smooth), or directly on function derivatives. Porting the code to a non-smooth optimization routine, while the problem is dominantly smooth, is very likely not the best option.

Manuscript received November 7, 2012; revised Xxxxxxx 00, 201X. This work was supported by XX.

A. Escande is with the CNRS-AIST JRL, Japan; S. Miossec is with PRISME-Univ. Orléans, France; M. Benallegue is with the LPPA, College de France, France; A. Kheddar is with the CNRS-AIST JRL, Japan and the CNRS-UM2 LIRMM, France.

To still use smooth methods when they are beneficial, the idea is to enforce the continuous differentiability of the distance function. This is the subject of this paper. In [1], the problem is formulated in 2D using a rectangle representing the shape of a 3 degrees-of-freedom (DOF) robot moving among fixed 2D convex obstacles. To add continuously differentiable collision avoidance constraints between the convex shapes of the robot and the obstacle, Gilbert and Johnson suggested the idea of lightly "bulging" the rectangle approximating the shape of the robot so as to enforce the strict convexity of its shape. *Thus far, the automation of the "bulging" and its extension to the 3D case have still not been addressed: this is the core contribution of our work.*

In this paper, we significantly revisit our work in [2] and propose a solution for the 3D case: an automated and robust geometrical operator to construct a strictly convex shape of a given initial geometrical model. In simple words, we present a method to *bulge* (i.e., round or curve) a polyhedral convex hull into a strictly convex hull/shape. Besides, our operator provides tunable bulging, ranging from the sphere bounding volume (i.e., the sphere with minimum radius encapsulating the entire object) to a polyhedral convex hull (i.e., the convex polyhedron with minimum volume that bounds the entire object). Our operator builds like a polyhedral convex hull, except that vertices are patches from small spheres of predefined radius, faces are patches from predefined bigger-radius spheres, and edges are patches from tori connecting adjacent features: therefore we named it Sphere Tori Patches (STP), and since it is a bounding volume, we added BV, yielding the name STP-BV.

Our second contribution is a study of the impact of STP-BVs when used for distance computation in optimization problems. We take three examples of increasing complexity to study their usefulness in terms of convergence success, speed and precision. We compare (i) two smooth solvers without STP-BVs distance regularization, (ii) the same two solvers with STP-BV, (iii) a non-smooth solver without STP-BV. The results show that the second option outperforms the others. While the last example is inspired by a real-life robotic application, we do not discuss here how to use optimization in robotics. Examples of the use of STP-BVs can be found in [4] for planning, [5] for trajectory optimization, and [6] [7] for control.

Many robotics research problems are based on distance computation: motion planning, grasping, simulation, collision-avoidance control, and design based on collision-free workspace computation. Among them, smooth distance computation is needed for smooth optimization, often used

in motion planning and robot design, and for stable control, which could then benefit from the use of STP-BV.

The rest of this paper is organized as follows: We first review related works (Section II); then, we study a 2D example featuring the problem of the distance gradient discontinuity in optimization (Section III); we highlight additional mathematical properties of the distance function (Section IV) and formally introduce the STP-BV (Section V). Section VI tackles the construction of the tori and spheres, and then, an algorithm to build an STP-BV from a point cloud is described in Section VII. Finally, we use our STP-BV in a simple 3D example and a more complex robotic one (Section VIII), where the distance is computed with the algorithm described in [8].

## II. Related Works

There are many types of strictly convex shapes. Used as bounding volumes (covering at best the robot's links), they would guarantee the differentiability of the distance function. However, they either have a poor volume ratio or are time consuming when used in distance computations. Spheres, ellipsoids, and super-ellipsoids are such strictly convex shapes. A single sphere or ellipsoid per link exhibit a weak volume ratio fitting, particularly for relatively thin and long robot links. The use of parts of these shapes to better fit each link increases the number of constraints to be added in the optimization problem. By combining several spheres, $k$-IOS [9] can fit the object tighter, yet the ratio might be no better than 0.5 in pathological cases. Super-ellipsoids or even hyper-quadrics are also very good candidates, but computing distance from these shapes to others is too time consuming [10] to be considered in control. Further, the volume ratio fitting for a general convex shape cannot be made arbitrarily close to 1.

Few other papers addressed the non-differentiability of the distance between convex bodies. Rusaw in [11] used non-smooth analysis in the context of sensory-based planning. In [12], the authors used cylinders as BVs to cover a robot, and acknowledged problems in the control when pairs of cylinders become parallel. They proposed to solve this issue by thresholding the output speed of the robot. Recently, many authors acknowledged this problem when using capsules [13] or convex hulls in optimization problems [14].

To avoid collisions, pseudo-distances can also be used, provided they have good differentiability properties. Distance fields such as in [15] are a possible solution. However, they remain computationally expensive as stated in [16] and need to be adapted for a pair of moving objects. Closer to the Euclidean distance, Zhu et al. [17] study the differentiability of a family of pseudo-distances based on the gauge function of a convex polygon (thus encompassing the $L_1$ and $L_\infty$ distances). These pseudo-distances are differentiable almost everywhere in the same manner as the Euclidean distance. However, the sufficient condition for differentiability is not met in the same cases as those for the Euclidean distance as well as for additional cases (when a face of the Minkowski difference of the two objects is parallel to a face of the polygon defining the pseudo-distance). The authors in [17] do not provide a way to avoid these cases.

## III. Motivation

### A. Example of the distance gradient's discontinuity in 2D

Let us consider the segment depicted in Fig. 1. $d$ is the distance from this segment to the horizontal axis, and we look at the gradient of $d$ with respect to $\theta$. This gradient (see Fig. 3(a)) is shown to be discontinuous in [11]. The discontinuity occurs when the segment becomes parallel to the horizontal axis: in this case, the witness point of the segment switches instantly between its two endpoints (see Fig. 2). This leads to a gradient discontinuity of $d$ since both endpoints move in an opposite direction when $\theta$ changes. Such a point of discontinuity is named *kink point*.

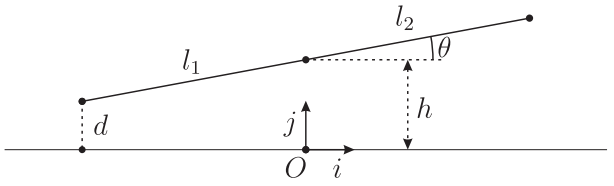
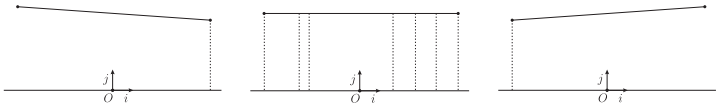
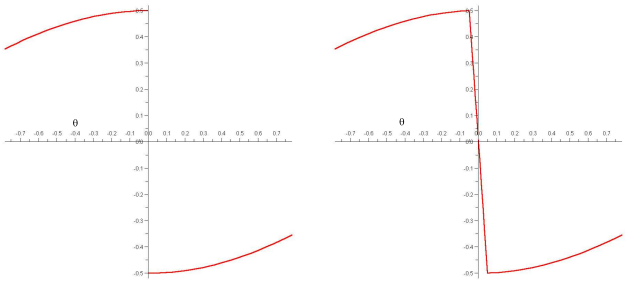



Fig. 1. A segment free to rotate around one of its points. $d$ denotes the distance to the horizontal axis.



(a) $\theta < 0$ (b) parallel case: $\theta = 0$ (c) $\theta > 0$

Fig. 2. When $\theta$ passes over 0 2(b), the witness points jump between the configuration of 2(a) and 2(c)



(a) gradient discontinuity (Fig. 1) (b) regularized gradient (Fig. 4)

Fig. 3. Gradient $\partial d/\partial\theta$ of the distance between the bar (left) or its bulged version (see Fig. 4) (right) and the horizontal axis. In the first case the gradient is discontinuous at $\theta = 0$ while in the second case it varies quickly but continuously around $\theta = 0$

### B. Regularization of the gradient

To avoid this gradient discontinuity, the idea is to prevent jumps of the pair of witness points. This case occurs when *flat* parts of both objects' shapes become parallel. We remove the flat parts by bulging (i.e., curving) them. For instance, we replace the segment of our example by a part of a circle having a large radius (see Fig. 4). The now unique witness point travels continuously between both endpoints and the gradient becomes continuous (see Fig. 3(b)). This is the main idea behind the bounding volume that we describe in

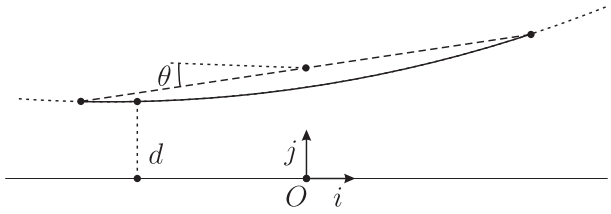



Fig. 4. The segment is replaced by a part of circle with a big radius. Now, when going over $\theta = 0$, the pair of witness points moves continuously

Section V. Subsequently, bulging an object is called *gradient regularization* or *regularization* for short.

### C. Efficiency of the regularization

Let us consider the following optimization problem

$$\min_{h,\theta} \quad h^2 + (\theta - \theta_b)^2 \tag{1}$$

$$\text{s.t.} \quad d(h,\theta) \geq d_0 \tag{2}$$

where $h$ denotes the height of a 1m bar center ($l_1 = l_2 = 0.5$m in Fig. 1); $\theta$, its orientation; $d_0 > 0$, a margin distance; and $\theta_b$, the target orientation. We can show that this optimization problem converges to the kink point $(h, \theta) = (d_0, 0)$ for $-d_0 l_1 \leq \theta_b \leq d_0 l_2$. In the other cases the solution is a regular point. We compare several strategies to solve this problem (and the other ones discussed in this paper): using smooth optimization routines with or without regularization, and using non-smooth routines without regularization. For the former, we use two solvers, FSQP [18][1] and NPSOL[2]. These are SQP algorithms, a class of routines among the fastest for nonlinear smooth problems, with a superlinear convergence. For the latter, we chose SolvOpt [19][3]. Analytic Center Cutting Plan Methods (ACCPM), used for example by the optimization solver OBOE [20][4], deal with non-smooth problems as well, but require the optimization problem to be convex, whereas a distance function provides a non-convex constraint.

Each solver has its own parameters for stopping criteria. Thus, we compare them by the number of iterations or computation time needed to obtain a desired precision of the solution (determined analytically when possible or by brute force). We consider two possibilities by choosing $\theta_b$ appropriately: convergence to a regular point or to a kink point. Each case study is optimized 100 times with the same random initial conditions, and the results are quantified as mean values. Computations are performed for different regularization radii (when applicable) and different algorithm parameter values. For a fair comparison, the best results in terms of computation time with respect to (wrt) the solution precision are kept. This corresponds to having tuned the solvers to obtain the best of their capabilities for the problem. Results are presented in Fig. 5. Using Tab. I, one can also obtain the mean results in terms of the computation time. All the computation times given in this paper are obtained on an Intel Core 2 Duo P8600 (3M Cache, 2.40 GHz).

| SolvOpt | FSQP | NPSOL |
|---|---|---|
| 1.3 ± 0.1 | 10.4 ± 0.4 | 39.3 ± 7.4 |

TABLE I
MEAN TIME PER ITERATION IN $\mu s$ (WITH STANDARD DEVIATION) FOR OPTIMIZATION PROBLEM (1)-(2)

Notice the slower convergence to a kink point than to a regular one for all algorithms (except SolvOpt). For NPSOL, regularization is necessary to obtain convergence. For FSQP, surprisingly, convergence is obtained without regularization when using the Armijo line-search option. However, the regularization improves slightly the convergence time. We observed that regularization allows approximately $20\%$ less collision constraint evaluations. We have no explanation for the good behavior of FSQP for the non-smooth case. It may be attributed to the special line-search designed to maintain feasible iterates. It would be worth studying this issue from the optimization viewpoint.

Overall, the solver for non-smooth problems SolvOpt performs worse in terms of the number of iterations but better in terms of computation time than other solvers, because of a smaller computation time per iteration.

These results depend on the optimization problem (see others in Section VIII). We later see that the relative performance of SolvOpt degrades quickly with more complex problems.

This study also suggests an optimal choice of the regularization radius wrt the desired precision, see Fig. 5(c). The precision for a given radius depends on the body's shape: bodies with smaller edges and faces give a better precision for the same regularization radius. For a 1m-bar, to obtain a $10^{-n}$ precision of $(h, \theta)$, the best choice is a circle of radius $10^n$m. From the study of the relation between the obtained precision on $h$ and the sphere radius, we observed that the radius providing the fastest convergence is linked to the regularization precision given by Theorem 6.2. Indeed, smaller radii give less precision (bigger bulging), but bigger radii result in stiffer problems, whose convergence is slower. Hence, the best choice of regularization radius is the smallest possible radius that fulfills the desired precision.

## IV. DISTANCE CONTINUITY PROPERTIES

Gilbert and Johnson [1] demonstrated the necessity of having one body strictly convex and the other convex to obtain the differentiability of their separating distance function. We recall this theorem and complement it with additional properties.

### A. Problem definition and notations

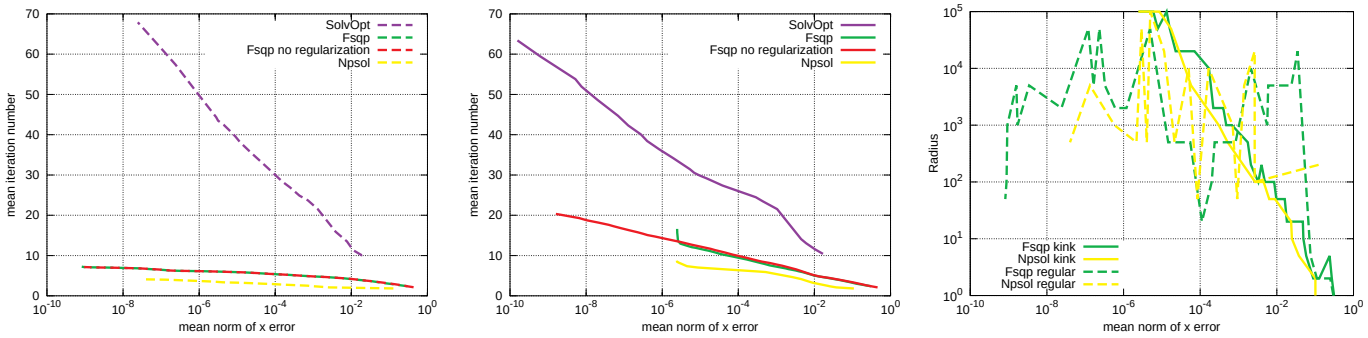
*Definition 4.1 (Convexity and strict convexity):* An object $O$ is convex iff $\forall (X, Y) \in O^2$ and $\forall t \in \,]0, 1[$ the point $P$ defined by $P = (1-t)X + tY$ lies in $O$. Furthermore, it is said to be *strictly* convex if $X \neq Y$ and $P$ lies in *int* $O$, the interior of $O$.

[1] http://aemdesign.com/

[2] http://www.sbsi-sol-optimize.com/asp/sol_product_npsol.htm

[3] http://www.uni-graz.at/imawww/kuntsevich/solvopt/

[4] https://projects.coin-or.org/OBOE/




(a) Convergence to a regular point (b) Convergence to a kink point (c) Best radius for a given precision

Fig. 5. Comparison of the number of iterations (ordinates) required to achieve a desired precision of the result (abscissa with $x = [h, \theta]$) for the optimization problem (1)-(2) with $d_0 = 0.05$ and $\theta_b = 0.01$ (Fig. a) or $\theta_b = 0.02$ (Fig. b), and corresponding regularization radius (Fig. c). One can note that when converging to a regular point, FSQP behaves exactly the same with or without regularization. Furthermore, there is no real relation between the optimal radius and the precision when converging to a regular point, which is normal since regularization is not an issue at this point.

We consider the distance between two convex objects $O_1$ and $O_2$. This distance is denoted by $\delta$. The relative position between the two objects is parameterized by 6 scalars (3 for rotation and 3 for translation). $q$ denotes the vector made of these scalars.

We call *witness points* a pair of points of $O_1 \times O_2$ that are at the distance $\delta$. Under certain conditions (see later Lemma 4.1), this pair is unique. Thus we can define $p_{\min}(q) = (p_{1\min}(q), p_{2\min}(q))^T$ as the function that associates the pair of witness points to each $q$.

Further, the surface of each object can be described by a function of two parameters. Let $u$ be the 4-dimensional vector of these $2 \times 2$ parameters, and $r_1$ and $r_2$ be these functions for $O_1$ and $O_2$ (to simplify, we define $r_1$ and $r_2$ as a function from a subset of $\mathbb{R}^4$ to $\mathbb{R}^3$ even though both of them are functions from a subset of $\mathbb{R}^2$ to $\mathbb{R}^3$).

The witness points being on the objects' surfaces, we define $u_{\min}$ as the function of $q$ that returns the vector $u$ of these points. Let $R(q)$ be the rotation matrix and $T(q)$ the translation vector. Then, we have $p_{\min}(q) = (r_1(u_{\min}(q)), R(q)r_2(u_{\min}(q)) + T(q))^T$

Finally, we define $f(q, u) = r_1(u) - R(q)r_2(u) - T(q)$ as the distance vector between the two witness points parameterized by $u$, so that

$$\delta(q) = \min_u \|f(q, u))\| = \|f(q, u_{\min}(q))\| \tag{3}$$

Note that $f$ is $C^\infty$ with respect to $q$ since $R$ and $T$ are rotation and translation, and that if $r_1$ and $r_2$ are $C^k$ with respect to $u$, so is $f$.

### B. Strict convexity of a body

As shown in Section III, a gradient discontinuity occurs at a jump of the witness points. This happens around a configuration in which the witness pair is not unique, such as when an edge is parallel to a face. If one object is strictly convex, parallelism does not happen. The following successive lemmas and theorems demonstrated in [1] link the uniqueness of witness points (hence the continuity of the gradient) with the strict convexity of one object. Thus far, they are valid for $\delta > 0$.

*Lemma 4.1: (Uniqueness of witness points)* There is a unique pair of witness points if at least one of the bodies is strictly convex (and the other is convex).

*Lemma 4.2:* 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 and the other is convex ($u_{\min}$ and $p_{\min}$ are continuous functions of $q$).

*Theorem 4.3:* The minimum distance between two convex bodies is a $C^1$ function of $q$ if at least one of the bodies is strictly convex and the other is convex.

If the surfaces of these objects have additional continuity properties and $\frac{\partial^2 f^2}{\partial u^2}$ is invertible (which is always the case but for pathological cases), the distance function becomes smoother:

*Lemma 4.4:* If the surface of both bodies are $C^k$, with $k \geq 2$ and one body is strictly convex, then the witness points are $C^{k-1}$ functions 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 of the minimization problem in (3)) which can be rewritten as $\frac{\partial f^2}{\partial u}(q_0, u_0) = 0$.

Let us note that $F(q, u) = \frac{\partial f^2}{\partial u}(q, u)$. $F$ is $C^{k-1}$, $F(q_0, u_0) = 0$ and $\frac{\partial F}{\partial u}(q_0, u_0)$ is invertible. Thus $u$ is locally a $C^{k-1}$ function of $q$ (implicit functions theorem). Subsequently, $u_{\min}$ is a $C^{k-1}$ function of $q$. ■

*Theorem 4.5:* If the surfaces of both bodies are $C^k$, with $k \geq 2$ and one body is strictly convex, then the minimum distance between them is $C^k$.

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

$$\begin{aligned} \frac{\partial \delta}{\partial q}(q) &= \left(\frac{\partial f}{\partial q} + \frac{\partial u_{\min}}{\partial q} \cdot \frac{\partial f}{\partial u}\right)^T \frac{f}{\|f\|} \\ &= \left(\frac{\partial f}{\partial q}(q, u_{\min}(q))\right)^T \frac{f(q, u_{\min}(q))}{\|f(q, u_{\min}(q))\|} \end{aligned} \tag{4}$$

since $\left(\frac{\partial f}{\partial u}(q, u_{\min}(q))\right)^T f(q, u_{\min}(q)) = 0$

Using Lemma 4.4, the demonstration is straightforward. ■

To sum up, the continuity properties of the minimum distance function with respect to the relative position of two convex bodies are as follows:

- 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 and $C^0$ surfaces ensures the $C^1$ property of the distance. Having $C^1$ surfaces does not improve the continuity of the distance as compared to having $C^0$ surfaces,
- Additional $C^k$ property of both surfaces yields $C^k$ smoothness of the distance if $\frac{\partial^2 f^2}{\partial u^2}$ is invertible.

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

### C. Interpenetration case

The $C^n$ property for $n > 0$ cannot be reached everywhere in the interpenetration case: the distance minimization problem is not convex anymore. Indeed, in some configurations there are several (up to infinite, see Fig. 6) pairs of witness points. Jumps between witness pairs are thus inevitable resulting in gradient discontinuities. However, we can retain the results

Fig. 6. For two objects in interpenetration (one in plain line, on in dash line), a degenerate case where there is an infinity of witness pairs, some of which are depicted in gray.

of the abovementioned theorems for a subset of penetration cases. But first, we need to ensure the continuity properties between the penetration and the non-penetration case. 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 with opposite normal vectors (which is the same definition as in the non penetration case). This definition is equivalent to the minimum separating translation distance (see [21], Chapter 2). Jumps between pairs do not happen if the penetration depth is less than twice the minimal radius of curvature of the penetrating parts of the objects. Under this assumption of "slight" penetration the previous results hold; if not we may then have gradient discontinuities. In the worst case, we may encounter the discontinuities for $\delta = 0$ when at least one witness point is on a vertex and the other on an edge. These are cases of bifurcation of the number of witness pairs. However, these discontinuities will be avoided when a security margin is considered for collision avoidance [1], which is the case for our bounding volume (see the next section). Therefore discontinuities will only occur in strict penetration cases, which are repulsive situations: the gradient points away from them. Hence they should not be a problem in most applications where penetration can occur (like in optimization starting with an unfeasible point).

## V. Sphere-Torus-Patch Bounding Volumes

### A. Definition

We consider a point cloud $\mathcal{P}$, and, for $r \geq 0$, the set $\mathcal{P}_r$ of (closed) balls $\mathfrak{B}(P, r)$ centered at each $P \in \mathcal{P}$ and with radius r ($\mathcal{P}_0 \equiv \mathcal{P}$). Let $\mathcal{B}_{R,r}(\mathcal{P})$ be the set of all balls with radius $R$ that encloses every ball of $\mathcal{P}_r$ ($R-r$ must be at least the radius of the smallest sphere containing $\mathcal{P}$).

*Definition 5.1 (STP-BV):* We name STP-BV of $\mathcal{P}$ for the radii $R$ and $r$ the intersection of all the balls of radius $R$ containing the balls of radius $r$ centered at each point of $\mathcal{P}$.

$$\mathrm{STP}_{R,r}(\mathcal{P}) = \bigcap_{B \in \mathcal{B}_{R,r}(\mathcal{P})} B \tag{5}$$

Henceforth, we ignore the special case where there is a single point in $\mathcal{P}$ and $r = 0$ (therefore, the STP-BV has a non-empty interior).

### B. Strict convexity

We show now that an STP-BV is strictly convex, which is not straightforward. Indeed, the intersection of an infinite number of convex volumes is always convex, but the intersection of an infinite number of strictly convex volumes is not necessarily strictly convex. We prove with the following lemma and theorem that the strict convexity is retained in the case of STP-BV. The lemma shows that for a point strictly on a segment, a minimal distance to the boundary of any ball with radius $R$ containing this segment can be guaranted.

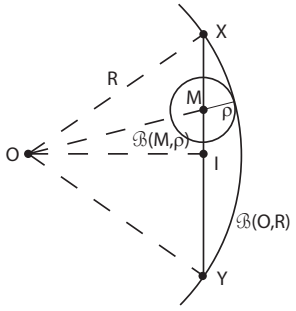



Fig. 7. Figure for Lemma 5.1.

*Lemma 5.1:* Let $X$ and $Y$ be two points such that $X \neq Y$, and $XY \leq 2R$. Let $t \in \,]0, 1[$ and define $M = (1-t)X + tY$ and $\rho = R - \sqrt{R^2 - t(1-t)XY^2}$. Then, for any ball $\mathfrak{B}(O, R)$ that contains $X$ and $Y$, *int* $\mathfrak{B}(M, \rho) \subset$ *int* $\mathfrak{B}(O, R)$ and *int* $\mathfrak{B}(M, \rho)$ is not empty.

*Proof:* $O$ being not fixed, the longest distance between $O$ and $M$ is achieved when $O$ is such that $X$ and $Y$ are both on the surface of $\mathfrak{B}(O, R)$: $OX = OY = R$ [5] . We consider this case and we denote the middle point of the segment $[X, Y]$ as $I$ (see Fig. 7). The triangle OIX is orthogonal at $I$ so that $OI^2 =$

[5]this is quite intuitive, but it can be proven by writing the maximization of $OM$ wrt $O$, with the constraints $OX \leq R$ and $OY \leq R$

$R^2-\frac{1}{4}XY^2$. Then considering the triangle OIM, orthogonal at $I$, we have $OM^2 = R^2-t(1-t)XY^2$. We have $t(1-t)XY^2 > 0$ since $t \in ]\,0,1\,[$ and $X \neq Y$, so that $OM < R$.

In the general case (i.e., $OX \leq R$, $OY \leq R$), we have $OM \leq \sqrt{R^2 - t(1-t)XY^2} < R$ so that whatever $O$, $M$ is at least at a distance $\rho > 0$ of the boundary of $\mathfrak{B}(O,R)$. ■

*Theorem 5.2: (Strict convexity of STP-BV)* $\text{STP}_{R,r}(\mathcal{P})$ is strictly convex

*Proof:* Let us show that for any $X \neq Y$ of the STP-BV and any $t \in ]\,0,1\,[$, $(1-t)X + tY \in$ *int* $\text{STP}_{R,r}(\mathcal{P})$.
Let $X \neq Y$. If $X, Y \in \text{STP}_{R,r}(\mathcal{P})$ then for every B of $\mathcal{B}_{R,r}(\mathcal{P}), X, Y \in B$. In particular, $XY \leq 2R$.
We define $t$, $M$ and $\rho$ as in Lemma 5.1. Then $\forall B \in \mathcal{B}_{R,r}(\mathcal{P})$, *int* $\mathfrak{B}(M,\rho) \subset$ *int* $B$ thus *int* $\mathfrak{B}(M,\rho) \subseteq$ *int* $\text{STP}_{R,r}(\mathcal{P})$ so that $M \in$ *int* $\text{STP}_{R,r}(\mathcal{P})$. ■

### C. Surface

Here, we describe the surface of an STP-BV. The following theorem helps us reducing the amount of work:

*Theorem 5.3: (Dilation)* $\text{STP}_{R,r}(\mathcal{P})$ is the dilation of $\text{STP}_{R-r,0}(\mathcal{P})$ by a ball of radius $r$.

*Proof:* This stems trivially from the fact that if $X$ is in a ball of radius $R-r$ then $\mathfrak{B}(X,r)$ is included in the ball having the same center, and radius $R$, and conversely. ■

For presenting how to build an STP-BV surface, we then take $r = 0$. For the sake of simplicity, we write $R$ instead of $R-r$.

A ball $B = \mathfrak{B}(C,R) \in \mathcal{B}_{R,0}(\mathcal{P})$ really limits the STP-BV if it is tangent to one or more points of $\mathcal{P}$. The following three cases are observed:

- $B$ is tangent to three or more points $P_i$ (see Fig. 9). The part of the corresponding sphere, which is inside the cone defined by the vectors $CP_i$ and with apex $C$, is a part of the boundary of the STP-BV.
- $B$ is tangent to two points $P_1$ and $P_2$. The geodesic arc on $B$ from $P_1$ to $P_2$ is part of the boundary of the STP-BV. Let $B_1$ be a ball tangent to $P_1$, $P_2$ and an additional point $P_3$. All the balls of $\mathcal{B}_{R,r}(\mathcal{P})$ tangent to $P_1$ and $P_2$ are obtained by rotating $B_1$ around $P_1P_2$ and away from $P_3$ until it becomes tangent to a new point $P_4$. The corresponding geodesic arcs form a part of torus (see Fig. 10), whose limits are the planes $C_1P_1P_2$ and $C_2P_1P_2$ where $C_1$ denotes the center of $B_1$ and $C_2$ denotes the center of the ball tangent to $P_1$, $P_2$ and $P_4$.
- $B$ is tangent to only one point. It limits the STP-BV only at this point.

The surface of the STP-BV is thus made of parts of spheres and tori, hence the name, Sphere-Torus-Patch Bounding Volume. The sphere parts can be seen as the "faces" of the volume and the tori as its "edges". The vertices are the points of $\mathcal{P}$ that define the axes of the tori. When $r > 0$, the vertices become parts of small spheres with radius $r$, see Fig. 8[6].

By definition, $\text{STP}_{R,0}(\mathcal{P})$ is the smallest bounding volume built with balls of radius $R$. $\mathcal{H}(\mathcal{P})$ being the convex hull of $\mathcal{P}$, we have $\mathcal{H}(\mathcal{P}) \subset \text{STP}_{R,0}(\mathcal{P})$. The parameter $R$ denotes the maximal radius of curvature of the STP-BV and controls the boundary on the maximal margin between the STP-BV and $\mathcal{H}(\mathcal{P})$, as shown in Theorem 6.2. When $R$ tends to infinity, $\text{STP}_{R,0}(\mathcal{P})$ tends to $\mathcal{H}(\mathcal{P})$. The STP-BV can thus be made arbitrarily close to a convex polyhedron. When used, the parameter $r$ gives the minimal distance between $\text{STP}_{R,r}(\mathcal{P})$ and $\mathcal{H}(\mathcal{P})$. It is used as a security margin. Both $R$ and $r$ are set by the user.

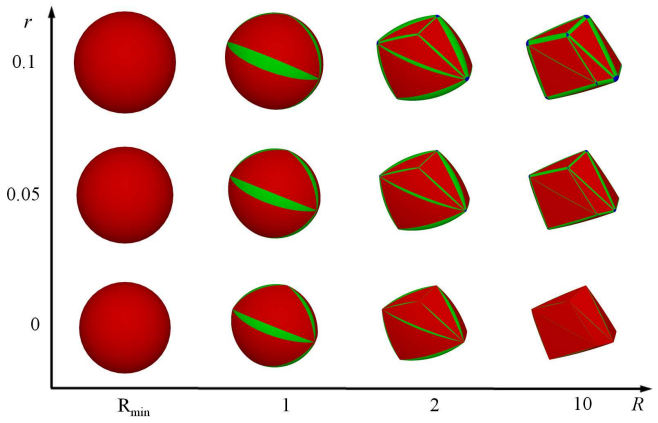



Fig. 8. STP-BV bulge tuning with the two radii $(R,r)$. The parts of big spheres are depicted in red, the tori in green and the small spheres in blue. $R_{\min}$ denotes the radius of the smallest bounding sphere. In this example, it is $0.87 + r$. For $(R,r) \to (\infty, 0)$ the STP-BV tends to the convex hull of the original polyhedron (bottom right). An increase in $r$ has a dilating effect whereas a decrease in $R$ from the convex hull $(\infty, 0)$ to the enclosing sphere $(R_{\min}, 0)$ achieves continuous bulging.

## VI. Geometric Constructions

The vertices and torus axes of $\text{STP}_{R,0}(\mathcal{P})$ define a polyhedron $\mathcal{H}_R(\mathcal{P})$ (usually but not necessarily convex). To each $B \in \mathcal{B}_{R,0}(\mathcal{P})$ tangent to three or more points of $\mathcal{P}$ corresponds a polygonal face of $\mathcal{H}_R(\mathcal{P})$ defined by these points (the face may not be planar in the rare case of non-planar cospherical points). If a face has more than three points, we divide it into triangles. The new edges correspond to flat tori. Obviously, $\mathcal{H}_R(\mathcal{P}) \subseteq \mathcal{H}(\mathcal{P})$.
In this section, we show how to build spheres and tori from edges and faces and prove some properties.

### A. Sphere construction

We consider a triangular face $\mathcal{H}_R(\mathcal{P})$ (see Fig. 9). $P_1$, $P_2$ and $P_3$ are its vertices given counterclockwise around the outer normal.

We want to build the (unique) sphere of center $C$ and radius $R$ that goes through the three vertices of the face, and is *above* the face (direction given by the outer normal).
Let us 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

[6]You can download the MS-OS demo software used for generating these figures from `https://sites.google.com/site/adrienescandehomepage/demo.rar`

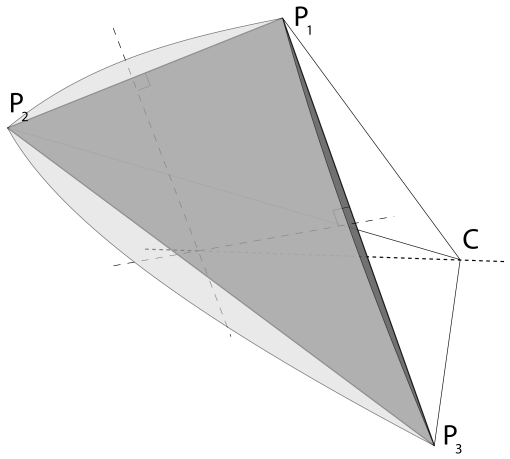



Fig. 9. Sphere construction: C is equidistant from each $P_i$ which means its projection is the circumcenter of the face (at the intersection of the edge); perpendicular bisectors are depicted by thin dash lines.

system to find the coordinates of $C$ in the frame $(P_1, u, v, w)$:

$$\begin{cases} u.c &= u^2/2 & \text{(median plan of } P_1P_2) \\ v.c &= v^2/2 & \text{(median plan of } P_1P_3) \\ c^2 &= R^2 & \\ c.w &< 0 & \text{(inner solution)} \end{cases} \quad (6)$$

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

$$\alpha = \frac{u^2v^2 - u.v \cdot v^2}{2(u \times v)^2}, \ \beta = \frac{u^2v^2 - u.v \cdot u^2}{2(u \times v)^2}$$

These are the coordinates of the circumcenter of the triangle. Replacing $\alpha$ and $\beta$ in the third equation leads to an equation of degree 2 in $\gamma$. Only one of its two solutions complies with the fourth equation: $\gamma = -\sqrt{\frac{R^2 - \alpha^2u^2 - \beta^2v^2 - 2\alpha\beta u.v}{(u \times v)^2}}$
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. Torus construction

We obtain the torus above one edge of $\mathcal{H}_R(\mathcal{P})$ by rotating the sphere of an incident face around this edge and keeping the resulting inner volume. The construction, depicted in Fig. 10, is based on the following result:

*Lemma 6.1:* The distance between the center of a sphere corresponding to a face and the median point $I$ of one of its edges depends only of the length $l$ of this edge and is $\sqrt{R^2 - \frac{l^2}{4}}$

*Proof:* With the notation of Fig. 10, it is the direct result of the Pythagorean theorem written for the triangle $(IC_1P_2)$. ■

The centers $C_1$ and $C_2$ of the two spheres corresponding to the two incident faces of the edge that we consider are thus on a same circle with center $I$ and radius $\sqrt{R^2 - \frac{l^2}{4}}$.
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

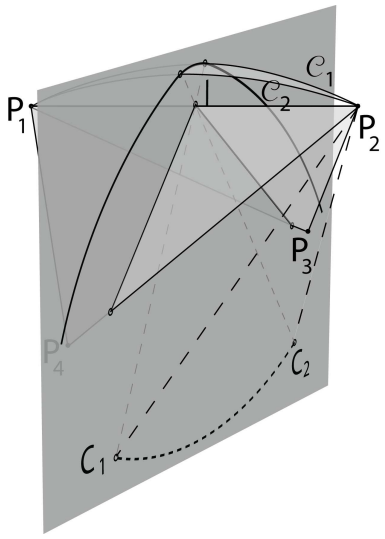



Fig. 10. Torus construction: the gray plane indicates the median plane of the edge $P_1P_2$, on which are depicted its intersection with the incident faces, the corresponding spheres, and the torus in between. The circles $\mathcal{C}_1$ and $\mathcal{C}_2$ denote the limits between the torus and the spheres (not depicted here) and generate this torus by rotation around $(P_1P_2)$.

coincides with the sphere centered in $C_1$.
By making this circle 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. Note that the torus does not have the usual shape of a donut since its usual small radius is bigger than the (usual) big one. The part that we consider here is on the inner side of the entire torus, as shown in white in Fig. 11. The torus

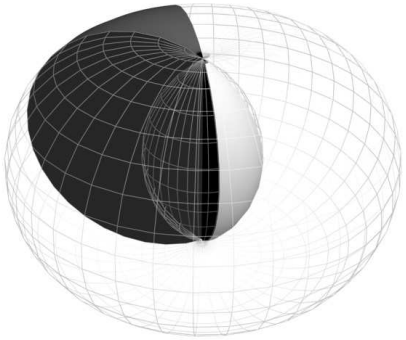


Fig. 11. The black and white parts are the portions of torus that we consider; the white surface is the part used in the STP-BV

and the spheres coincide in their delimiting planes and are perpendicular to these planes. Therefore, the torus is tangent to both spheres, and the junction between the torus and each sphere is then $C^1$.

### C. Properties

*Theorem 6.2: (Maximal margin)* If $a$ denotes the length of the longest edge of the polyhedron, the maximal margin between $\mathcal{H}_R(\mathcal{P})$ and $\text{STP}_{R,0}(\mathcal{P})$ is $R - \sqrt{R^2 - \frac{a^2}{12}}$

*Proof:* Since each edge of $\mathcal{H}_R(\mathcal{P})$ is the revolution axis of its associated torus, the maximal margin is achieved in its median plane, and is equal to $R - \sqrt{R^2 - \frac{l^2}{4}}$, $l$ being the length of the edge.
For a face, the following two cases are observerd with respect to the position of the circumcenter $H$:

- $H$ is outside of the face, and then, the maximal margin 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 margin is achieved when the face is equilateral and is $R - \sqrt{R^2 - \frac{a^2}{12}}$.

■

Since $\mathcal{H}_R(\mathcal{P}) \subseteq \mathcal{H}(\mathcal{P})$, this is also the maximal margin to $\mathcal{H}(\mathcal{P})$.

*Theorem 6.3: (Continuity property of STP-BV)* When $r > 0$, the surface of the STP-BV is $C^1$.

*Proof:* (sketch) For $r = 0$ the STP-BV is convex and its surface is $C^1$ everywhere but at its vertices. A dilation of $r > 0$ makes it round at the vertices and thus $C^1$ everywhere. ■

An STP-BV is even piecewise $C^\infty$ since the tori and the spheres are $C^\infty$ surfaces. However, as stated in Theorem 4.5, there is no advantage in having $C^1$ surfaces instead of $C^0$ with respect to the distance continuity.

## VII. STP-BV Construction

**Algorithm:** STP-BV construction: Pseudo-code

**Data**: point cloud $\mathcal{P}$, value of $R$
**Result**: the STP-BV polyhedron $\mathcal{H}_R(\mathcal{P})$
-$v$,$v_1$,$v_2$ and $v_3$ *are vertices*
-$c$, $c_1$ and $c_2$ *are sphere centers*
-next: *is a list of sets* $(v_1, v_2, v_3, (v, a))$*, sorted by increasing value of* $a$.
-output: *a structure describing* $\mathcal{H}_R(\mathcal{P})$

**BuildVolume**($\mathcal{P}$, $R$)
**begin**
  *init*(next, output)
  **while** *no next.isEmpty()* **do**
    $(v_1, v_2, v_3, (v, a)) \leftarrow$ next.pop()
    output.addFace($v_1$, $v$, $v_2$))
    **if** *next.*contains*($v_1$, $v$)* **then**
      next.*delete*($v_1$, $v$)
    **else**
      next.*insert*($v_1$ $v$,$v_2$, *nextPoint*($v_1$ $v$,$v_2$))
    **if** *next.*contains*($v$, $v_2$)* **then**
      next.*delete*($v$, $v_2$)
    **else**
      next.*insert*($v$, $v_2$,$v_1$, *nextPoint*($v$, $v_2$,$v_1$))
  *return* output

**Algorithm 1:** STP-BV construction.

There is by definition a one-to-one relation between the spheres and tori of the $\text{STP}_{R,0}(\mathcal{P})$ and the faces and edges of $\mathcal{H}_R(\mathcal{P})$. The construction of $\mathcal{H}_R(\mathcal{P})$ from $\mathcal{P}$ is described in Algorithm 1. It behaves mostly as a *gift wrapping algorithm* with spheres instead of planes (see Fig 12[7]): We find an initial face whose associated sphere contains $\mathcal{P}$. We then rotate the sphere around the edges of the face until it reaches new points of $\mathcal{P}$. This defines new faces, and new edges around which to rotate.

**Algorithm:** intermediate functions: Pseudo-code

**init**(next, output)
**begin**
  **for** *each* $(v_1,v_2,v_3) \in \mathcal{P}^3$ *with* $v_i \neq v_j$ **do**
    $c \leftarrow$ *sphere*($v_1$,$v_2$,$v_3$,$R$)
    **if** $\mathcal{P} \subseteq \mathfrak{B}(c, R)$ **then**
      output.*addFace*($v_1$,$v_2$,$v_3$)
      next.insert($v_1$,$v_2$,$v_3$, *nextPoint*($v_1$,$v_2$,$v_3$))
      next.insert($v_2$,$v_3$,$v_1$, *nextPoint*($v_2$,$v_3$,$v_1$))
      next.insert($v_3$,$v_1$,$v_2$, *nextPoint*($v_3$,$v_1$,$v_2$))
      *return* SUCCESS
  *return* FAIL

**nextPoint**($v_1$,$v_2$,$v_3$)
**begin**
  **for** *each* $v \in \mathcal{P} - \{v_1, v_2\}$ **do**
    $c_2 \leftarrow$ *sphere*($v_1$, $v$, $v_2$, $R$)
    **if** $\mathcal{P} \subseteq \mathfrak{B}(c_2, R)$ **then**
      **if** $v = v_3$ **then**
        *return* ($v$,$\pi$)
      **else**
        $c_1 \leftarrow$ *sphere*($v_1$,$v_2$,$v_3$, $R$)
        *return* ($v$,*angle*($c_1$, $c_2$, $v_1$, $v_2$))

**Algorithm 2:** Construction main sub-functions.

The *init* function (Algorithm 2) finds an initial face. Since we can only perform one rotation at a time, we need to maintain a list of rotations to be completed. Maintaining this list *next* is the main task of Algorithm 1. For a reason that we provide later, we always perform the smallest of the rotations in this list first. An element of *next* is a set $(v_1, v_2, v_3, (v_4, a))$ encoding that the (positive) rotation of the sphere associated to the face $v_1v_2v_3$ around the edge $v_1v_2$ reaches the point $v_4$ after a rotation of angle $a$ (see Fig. 10, $a$ denotes the angle between $IC_1$ and $IC_2$).
The list *next* is ordered according to the angles $a$. It supports the classical operations *pop* and *insert*. The operation *contains*($v_1$, $v_2$) returns true if any element of *next* begins by ($v_1$, $v_2$) or ($v_2$, $v_1$). *delete*($v_1$, $v_2$) removes such an element.
*output* is a structure describing a polyhedron (sets of vertices, edges and faces). The function *addFace*($v_1$, $v_2$, $v_3$) updates it by adding the face $v_1v_2v_3$ (described counterclockwise) and the corresponding edges and vertices, if not already present.
*sphere*($v_1$, $v_2$, $v_3$, $R$) returns the center of the sphere with radius $R$ as computed in Section VI-A.
*angle*($c_1$, $c_2$, $v_1$, $v_2$) returns the angle between $ic_1$ and $ic_2$ where $i$ denotes the middle point of $v_1v_2$. Lastly, *nextPoint*($v_1$, $v_2$, $v_3$) looks for the first point $v$ encountered while rotating the sphere associated to $v_1v_2v_3$ around $v_1v_2$, and returns it

[7]See also the attached video

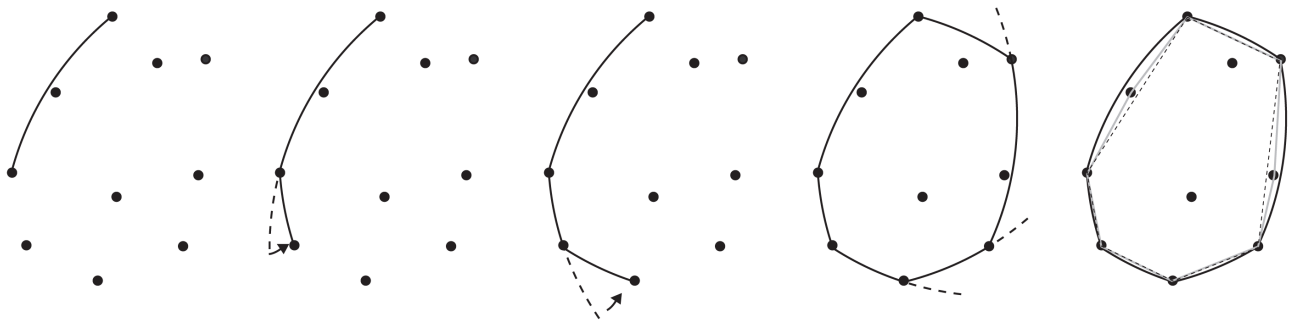


Fig. 12. Construction of the STP-BV of a cloud $\mathcal{P}$ in a 2d case by rotation of an initial circle (left) around the points as it encounters them. Some points of the convex hull $\mathcal{H}(\mathcal{P})$ (gray polygon) are dismissed, because of the curvature of the circle. The STP-BV polygon $\mathcal{H}_R(\mathcal{P})$ is depicted by a thin dashed line (right).

along with the rotation angle made to reach it. $v$ denotes the only point for which the sphere associated to $v_1vv_2$ contains $\mathcal{P}$, but when there are cospherical points, in which case any of these points will do. The special case where $v = v_3$ is discussed later.

At each iteration, Algorithm 1 processes a new face $v_1vv_2$ obtained by turning around $v_1v_2$. For the two edges $v_1v$ and $vv_2$ the following two cases can be observed:

- The edge was encountered before when processing a face $f$. There is no need to rotate around this edge anymore because rotating the sphere associated with $f$ around the edge would give $v_1vv_2$ and vice-versa. The rotation is removed from *next*.
- The edge was not encountered before. We need to perform the rotation around it later so it is added to *next*.

The output of Algorithm 1 is $\mathcal{H}_R(\mathcal{P})$ from which, with the computations described in the previous section, we can build the boundary of $\text{STP}_{R,0}(\mathcal{P})$ (spheres and tori with their limits) and thus any $\text{STP}_{R+r,r}(\mathcal{P})$.

The reason for choosing the smallest rotations first is to ensure the robustness of the algorithm wrt numerical errors in the case of polygonal faces whose vertices are on the same sphere (cospherical). 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 is selected. However this is sometimes not the case because of numerical rounding errors. Thus, different vertices can be chosen when we arrive on the face by turning around different edges, resulting most of the times in overlapping triangles, which leads to an algorithm failure. To avoid this, we force the algorithm to finish covering a polygonal face that it already began, by choosing to turn around the edge with the lowest rotation. Thus, we also avoid setting a threshold that defines when points are cospherical and searching for cospherical points each time there is a rotation around an edge.

In special cases, it can happen that the rotation of the face $v_1v_2v_3$ around $v_1v_2$ stops at $v = v_3$. This is for example the case when $\mathcal{P}$ has only three points, but it can also happen with more points[8]. In this case, after turning around $v_1v_2$, the algorithm will remove the rotations around $v_1v_3$ and $v_2v_3$ from *next* while one of them might still be needed. To avoid this, *nextPoint* forces the angle to $\pi$ when detecting the case, to ensure that the corresponding rotation will be the last one processed.

Finally, *init* fails in two cases: (i) if $R$ is smaller than the radius of $\mathcal{P}$ ($\text{STP}_{R,0}(\mathcal{P})$ does not exist), and (ii) in the case of thin long clouds where no ball of $\mathcal{B}_{R,0}(\mathcal{P})$ is tangent to three points of $\mathcal{P}$. In the latter case, $\text{STP}_{R,0}(\mathcal{P})$ is the torus whose axis is defined by the two furthest points of $\mathcal{P}$.

*Remark (computation time)*: For the polyhedrons that we usually consider, with a few hundred vertices, the construction time is around 1 s. The full STP-BV model the robot discussed in Section VIII-B, for example, is obtained in approximately 30 s.

## VIII. Applications

Distance computation algorithms can be categorized as feature-based and simplex-based algorithms. In [2], we implemented a featured-based distance computation for STP-BV based on V-Clip [22]. Later, see [8], we implemented a simplex-based method based on the GJK algorithm [23] [24] and its adaptation for the case of penetration distance [25]. Here, we use the latter solution because the penetration depth is necessary in optimization algorithms working with non-feasible iterates (such as a majority of the gradient-based algorithms like NPSOL).

In a nutshell, the integration of the STP-BV in a GJK scheme boils down to writing its support function $s$, i.e., the function that maps a vector $v$ to the extremal point $p$ of the STP-BV in the direction of $v$. Let $f$ be a part of a torus or sphere defining the surface of the STP-BV. All the normal vectors to $f$ form a cone $\mathcal{N}_f$. We have $s(v) = s_f(v)$ with $s_f$ the support function of the torus or sphere supporting $f$ such as $v \in \mathcal{N}_f$ (see [8] for more details). The support function is computed in $\mathcal{O}(\sqrt{n})$, where $n$ denotes the number of vertices in $\mathcal{H}_R(\mathcal{P})$. However, it comes close to $\mathcal{O}(1)$ when there is a temporal coherency.

The STP-BV library[9] developed consists of two parts: The STP-BV construction from a cloud of points, and the distance computation, which can be used like any classical distance computation algorithms. One can express collision avoidance constraints by requiring that the returned distance is positive,

[8]when for three points $v_1$, $v_2$ and $v_3$ both spheres built on $v_1v_2v_3$ and $v_1v_3v_2$ contain $\mathcal{P}$

[9]STP-BV library can be provided by authors upon request

| | SolvOpt | FSQP | FSQP no regul. | NPSOL |
|---|---|---|---|---|
| regular point | 35.9 ± 8.0 | 37.4 ± 1.8 | 33.0 ± 0.5 | 69.3 ± 27.6 |
| kink point | 39.6 ± 10.4 | 55.4 ± 4.8 | 60.5 ± 11.6 | 94.6 ± 30.4 |

TABLE II

MEAN TIME PER ITERATION IN $\mu s$ (WITH STANDARD DEVIATION) FOR OPTIMIZATION PROBLEM (7)-(8). STANDARD DEVIATION IS LARGE FOR NPSOL BECAUSE FIRST ITERATIONS TAKE MORE TIME.

or enforce contacts by imposing that the distance must be equal to zero.

### A. Cube problem

We study a generalization of problem (1)-(2) to the 3D case: Minimizing the distance of a 6 degrees of freedom cube to a reference configuration while avoiding a collision constraint with a fixed bigger cube, see Fig. 13. The fixed big cube is modeled as a polyhedron and the small one as a polyhedron or its STP-BV regularization. The optimization problem solved is as follows:

$$\min_{x} \quad \frac{\|C\|^2}{2} - \text{tr}(R(\psi_0,\theta_0,\phi_0)R(\psi,\theta,\phi)^T) \tag{7}$$

$$\text{s.t.} \quad d(x) \geq d_0 \tag{8}$$

where $x = [C_x, C_y, C_z, \psi, \theta, \phi]$ with $C = [C_x, C_y, C_z]^T$ as the cube center, and $\psi$, $\theta$, $\phi$ denote the roll, pitch, and yaw angles of the cube orientation, $\psi_0$, $\theta_0$, $\phi_0$ denote the reference orientation. $R(.,.,.)$ denotes a rotation matrix, tr(.) indicates for the trace, and $d(.)$ denotes the distance between the small and the big cubes. The minimum of $-\text{tr}(R(\psi_0,\theta_0,\phi_0)R(\psi,\theta,\phi)^T)$ is obtained when both rotation matrices are equal, which gives a value of $-3$. The computations performed are the same as with the bar problem, and we chose $(\psi_0,\theta_0,\phi_0)$ to make the optimization converge either to a regular point (Fig. 13, first case) or to a kink point (Fig. 13, third case). Similarly to the bar problem, despite the kink points of the problem being a subset of measure zero in the parameter space, the optimization will converge to them for regions of the $(\psi_0,\theta_0,\phi_0)$ space that have non-zero measure. The probability of converging to a kink point is thus not zero. Results are given in Fig. 14 and Tab. II. We did not run NPSOL without regularization because of the poor convergence obtained in the bar problem. One can observe that convergence is faster to a regular point. In terms of the computation time, SQP algorithms are faster than SolvOpt, whereas in terms of the number of iterations SQP algorithms are considerably better. In terms of the the computation time, the fastest convergence to a kink point is obtained with FSQP using STP-BV, followed by the use of NPSOL with STP-BV for high precision and FSQP with the cube (i.e. not using STP-BV) for low precision; in terms of the number of iterations, the fastest algorithm is NPSOL, followed by FSQP with STP-BV, and FSQP without STP-BV. At a regular point, in terms of the computation time, the fastest is FSQP without STP-BV, closely followed by FSQP with STP-BV and NPSOL; in terms of the number of iterations NPSOL is better, followed by FSQP. A noticeable difference with the bar case is that the use of STP-BV significantly improves the convergence speed of FSQP that can however correctly converge to a kink point. As in the bar case, the best choice of radius is the smallest one that gives the required precision.

### B. Collision-free humanoid posture generation

The work presented in this paper is originally motivated by the integration of collision avoidance in a planning algorithm presented in [4]. Part of the planning consists of generating postures. We show results from our posture generator using STP-BV for collision avoidance.

In brief, this posture generator performs an optimization under constraints such as equilibrium, required body positions (for example, robot-environment contacts, described as matches between frames of the robot and frames of the environment), and collision avoidance. The variable is the configuration vector. The criterion to be minimized can be any user-defined smooth function. In the following scenarios we use a simple one that gives fairly human-like postures for upright positions:

$$f(q) = \sum_{i=1}^{n_j} \left( q_i - \frac{q_i^{\min} + q_i^{\max}}{4} \right)^2 \tag{9}$$

where $n_j$ denotes the number of joints, and $q_i^{\min}$ and $q_i^{\max}$ represents the joint limits of $q_i$. The problem of collision avoidance in posture generation has also been addressed in [26] and in [27]. Authors in [26] generate postures from an optimization formulation. Spheres are used as coverage to define the collision avoidance constraints. The problem with their method is in the number of spheres required to cover the virtual avatar while maintaining a good precision of the shape, along with the combinatorial complexity of the sphere pairs to be considered as constraints. Authors in [27] 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. There are other approaches to the abovementioned problem such as in [28] where avoidance is made through potential fields. Fig. 15 shows the geometrical model of HRP-

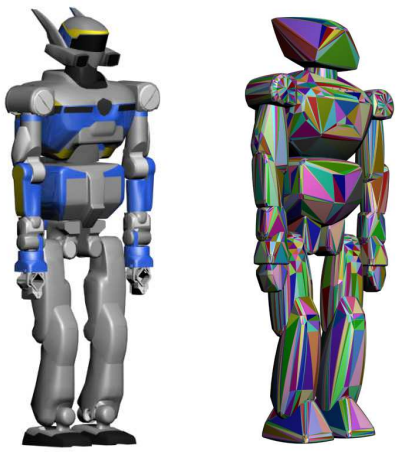


Fig. 15. HRP-2 robot and its STP-BV

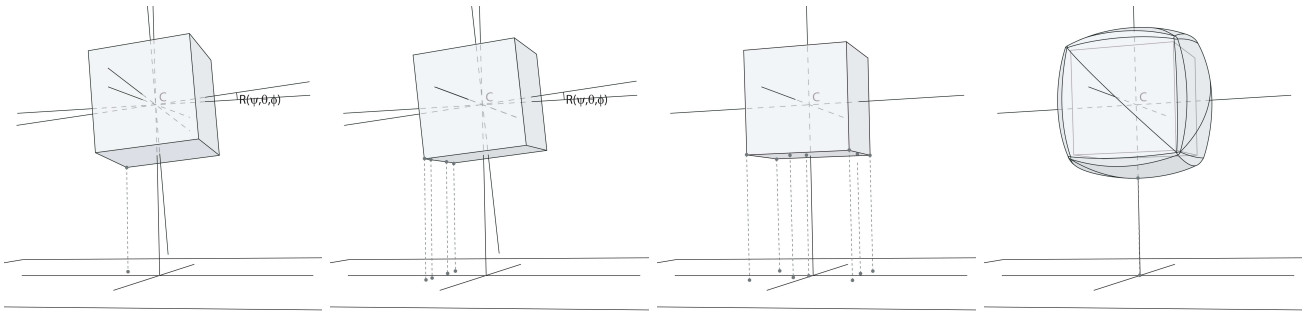
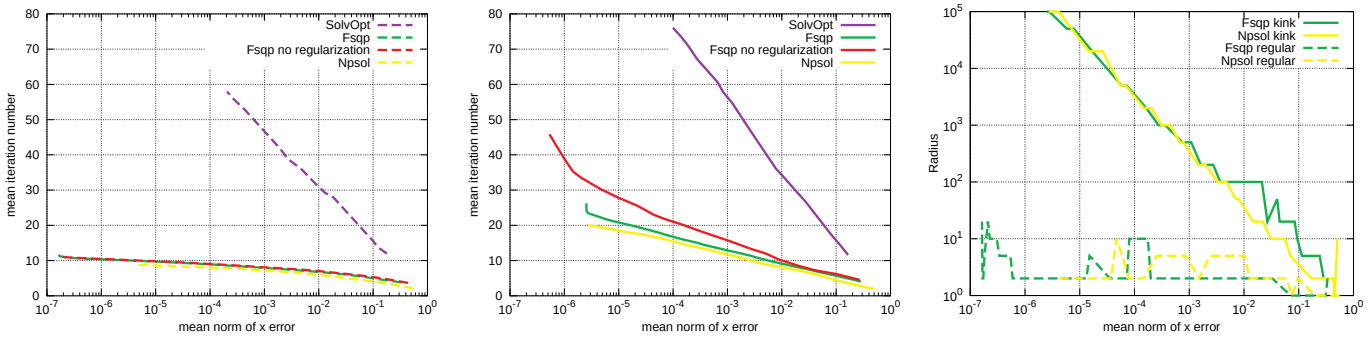


Fig. 13. The cube problem. From left to right: (a) case with only one pair of witness points, (b) one edge parallel to a face of the big cube, inducing an infinity of witness pairs, (c) parallel faces, (d) the STP-BV around the cube in the same configuration as (c). Only one pair of witness points is considered in this case.




(a) Convergence to a regular point (b) Convergence to a kink point (c) Best radius for a given precision

Fig. 14. Comparison of the computation time (ordinate) to achieve a desired precision of the result (abscissa) for the optimization problem (7)-(8), and the corresponding optimal regularization radius (Fig. c). Once again, FSQP behaves exactly the same with or without regularization when converging to a regular point (Fig. a).

2 and its STP-BV. The model of each HRP-2 body contains between 50 and 800 vertices. Parameters of STP-BV are as follows: $r = 2$cm and $R$ is fixed between 1m and $10^5$m. The environment is modeled with polyhedrons, as only one body of a pair needs to be strictly convex for the continuity of the distance gradient. Body pairs that need to be checked for self-collision have been studied in [29] and [30]. In the latter, look-up tables are used for dealing with composed joints, hence the safety margins of the bounding volumes do not restrain the movement possibilities. We focused on this point too, but since we need continuous gradients for all constraints, such a method was not possible. Thus, we used specific analytical functions of the joints values 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. There are 117 auto-collision constraints. Eight of them are analytical, the others being computed with the STP-BVs. We apply this posture generator to two scenarios (see an additional one in [2]):

*Enter-car scenario*: The robot has to place its right foot inside a car, on the floor in front of the driver seat, and its left foot on the floor outside of the car. The robot must be stable with only the left foot contact. Nineteen pairs are checked for collisions with the environment. The scenario is illustrated in Fig. 20.

*Reaching scenario*: HRP-2 attempts to reach an object near its waist. Both feet are placed on the ground, and the left hand position and orientation are fixed around the object. This scenario was designed to exhibit a non-smooth self-collision between the forearm and the flat part of the waist, as can be seen in the right picture of Fig. 21. For these two scenarios, we modified our protocol: We identify beforehand the log-linear relations between the parameters of the different algorithms and the radii. Each parameter $p_i$ is chosen to follow a law $p_i(R) = 10^{a_i \log_{10} R + b_i}$, where the $(a_i, b_i)$ have been identified to obtain the best iteration-precision ratio for a set of scenarios. We then run both scenarios above for $R$ varying from 1 to $10^5$; each time the parameters are given by the laws. Figs. 16-19 can then be seen as parametric functions of $R$.

We proceed this way to get closer to real-condition uses: We do not take the results for the best parameters and radii tuned for each particular scenario –such a tuning cannot be considered when generating thousands of different postures, e.g., in our case of planning. Each scenario is running with the same parameters, chosen to have the best compromise between the computation time and the expected precision of the solution. The initial postures are randomly chosen around the zero posture (Fig. 15) within a range of $4^\circ$ for the angles and 6.5cm for the root coordinates.

Figs. 16-19 and Tab. III present the computation results for the *Enter-car* and *Reaching* scenarios. In each case, the results are given with and without the use of STP-BV using NPSOL and FSQP (we did not run SolvOpt because of its poor results in the cube case). The displayed results are the mean values of 100 runs with random initial conditions, for

each value of $R$. Despite the short range of initial values, the algorithms sometimes converge to different local minima that we identified manually (for example, the left arm above the car door for the *Enter-car* scenario, instead of being along it as presented Fig. 20). For each of them, we compute the best parameter vector $x_{ref}$ over an extensive series of runs. The convergence precision is then measured as the distance between the obtained parameter vector $x$ of a run and its closest local minimum. At the optimum of the *Enter-car* scenario, 6 collision constraints with the environment are active. Only one of them is non-smooth at the solution. At the optimum of the *Reaching* scenario, 1 collision constraint is active (between the waist and the left arm) which is non-smooth at the solution. The convergence ratios are presented in Figs. 16 and 18, and for the cases which converged, Figs. 17 and 19 illustrate the convergence speeds.

For both scenarios, FSQP allows a better convergence ratio than NPSOL. Non-convergences are due to the impossibility of finding a feasible solution. On average, for NPSOL, the use of STP-BV decreases the number of iterations by approximately 50% and the computation time by approximately 25%. On average, for FSQP, the use of STP-BV does not significantly improve the number of iterations, and the computation time is slightly slower, but it allows reaching better mean precisions. For both FSQP and NPSOL, the confidence ellipses are smaller with STP-BV, representing a better precision reliability of the obtained solutions.

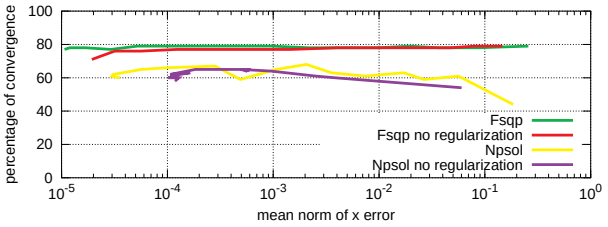



Fig. 16. *Enter-car scenario*: Percentage of convergence (ordinate) achieved for a desired precision of the result (abscissa).

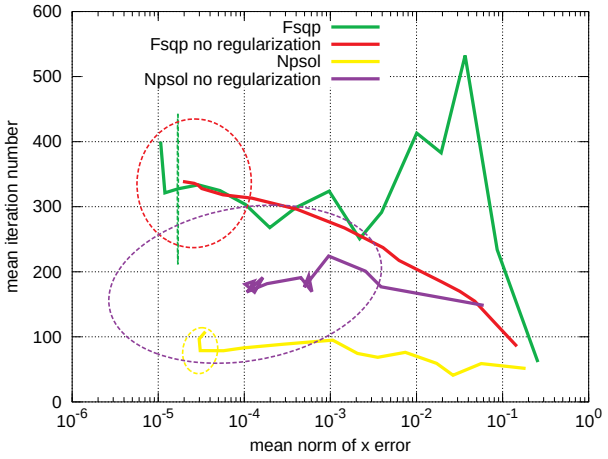



Fig. 17. *Enter-car scenario*: Comparison of the mean iteration number (ordinate) to achieve a desired precision of the result (abscissa). Confidence ellipses represent the repartition of solutions around an arbitrary mean point.

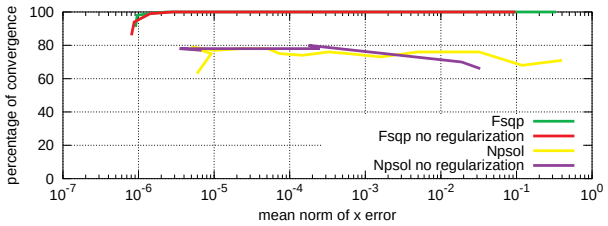



Fig. 18. *Reaching scenario*: Percentage of convergence (ordinate) achieved for a desired precision of the result (abscissa).

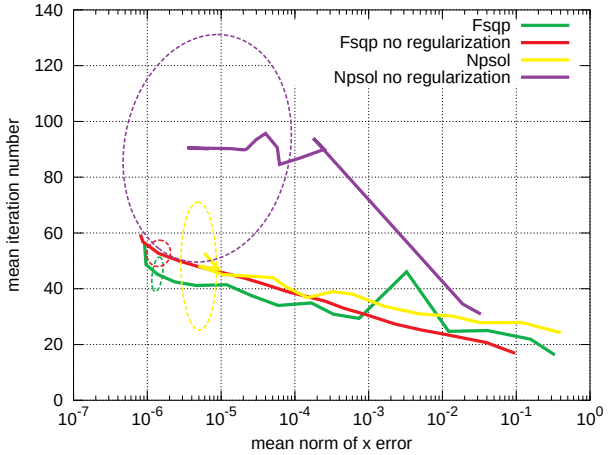



Fig. 19. *Reaching scenario*: Comparison of the mean iteration number (ordinate) to achieve a desired precision of the result (abscissa). Confidence ellipses represent the repartition of solutions around a mean point.

The use of STP-BV significantly improves the convergence speed of NPSOL because a non-smooth collision is active at the solution, which leads to convergence problems without STP-BV. However, as compared to the cube problem for which no convergence is obtained without STP-BV, here, convergence is obtained because the non-smoothness is relatively not dominant: indeed, the size of the face causing gradient discontinuity is considerably smaller than the cube's. For FSQP, we do not have explanations why the non-smoothness has so little impact on the convergence speed, except that it might have something to do with the feasibility of the iterates. Our interpretation of the non-convergence due to unfeasibility with NPSOL is that the nonlinear problems that we attempt to solve are highly non-convex, particularly because of the collision constraints. The reason why FSQP accommodates better with this non-convexity is because it keeps feasible iterates, what may help to avoid unfeasible local minima in which NPSOL falls. Another consequence of the non-

| | FSQP | FSQP no regul. | NPSOL | NPSOL no regul. |
|---|---|---|---|---|
| Enter-car | 508 ± 81 | 497 ± 103 | 369 ± 85 | 218 ± 17 |
| Reaching | 304 ± 32 | 231 ± 8 | 415 ± 120 | 267 ± 96 |

TABLE III

MEAN TIME PER ITERATION IN $ms$ (WITH STANDARD DEVIATION) FOR *Enter-car* AND *Reaching* SCENARIOS.

convexity of the problem is that in approximately 10-40% of the convergence cases with NPSOL, the convergence was obtained in other local minima.
Even if the problems considered are non-convex, we do not address the problem of a global minimum search in this paper.

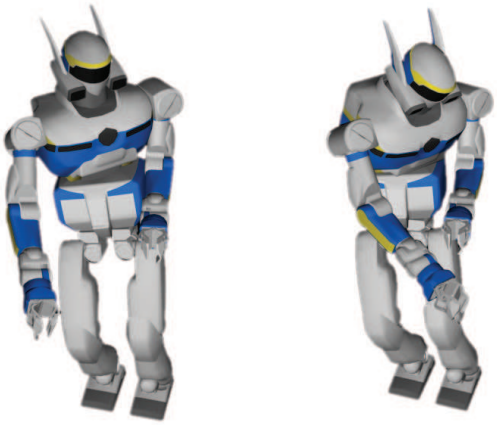


Fig. 21. Reaching scenario. Left: posture obtained without collision avoidance constraints. Right: successful collision-free posture.

## IX. Conclusion

We propose a new bounding volume operator to enforce the continuity of the gradient of the distance function. The main idea is to ensure the strict convexity of the bounding envelopes of the objects, which are computed off-line. We proposed such a bounding volume made as a patch of parts of spheres and tori, the STP-BV, whose parameter choice is a trade-off between the tightness of the volume and the regularization of the gradient of the distance. We showed that this regularization together with classical smooth optimization is an interesting alternative to the direct use of a non-smooth approach, through academic cases and a collision-free (including self-collision) optimization-based humanoid posture generation for HRP-2. The approach can also be of interest for any gradient-based control scheme.

## Acknowledgment

This work is partially supported by grants from the RoboHow.Cog FP7 `www.robohow.eu`, and by JSPS Grant-in-Aid for Scientific Research (B), 22300071, 2010.

## References

[1] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE Journal of Robotics and Automation*, vol. RA-1, no. 1, pp. 21–30, March 1985.

[2] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distances for humanoids free-collision optimized-postures generation," in *IEEE/RAS International Conference on Humanoid Robots*, Pittsburgh, USA, Nov 20-Dec 1 2007.

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

[4] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, 2013, to appear.

[5] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *International Journal of Robotics Research*, vol. 32, pp. 1104–1119, 2013.

[6] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Auton. Robots*, vol. 32, no. 4, pp. 433–454, May 2012.

[7] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar, "Real-time (self)-collision avoidance task on a HRP-2 humanoid robot," in *IEEE International Conference on Robotics and Automation, ICRA-08*, 2008.

[8] M. Benallegue, A. Escande, S. Miossec, and A. Kheddar, "Fast $C^1$ proximity queries using support mapping of sphere-torus-patches bounding volumes," in *IEEE International Conference on Robotics and Automation*, 2009.

[9] X. Zhang and Y. J. Kim, "$k$-IOS: Intersection of spheres for efficient proximity query," in *IEEE International Conference on Robotics and Automation*, Saint Paul, MN, May 14-18 2012, pp. 354–359.

[10] N. Chakraborty, J. Peng, S. Akella, and J. Mitchell, "Proximity queries between convex objects: An interior point approach for implicit surfaces," in *2006 IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006, pp. 1910–1916.

[11] S. Rusaw, "Sensor-based motion planning in SE(2) and SE(3) via nonsmooth analysis," Oxford University Computing Laboratory, Tech. Rep., September 27 2001.

[12] R. Patel, F. Shadpey, F. Ranjbaran, and J. Angeles, "A collision-avoidance scheme for redundant manipulators: theory and experiments," *Journal of Robotic Systems*, vol. 22, no. 12, pp. 737–757, 2005.

[13] A. Dietrich, T. Wimböck, A. Albu-Schäffer, and G. Hirzinger, "Integration of reactive, torque-based self-collision avoidance into a task hierarchy," *IEEE Transaction on Robotics*, vol. 28, no. 6, pp. 1278–1293, December 2012.

[14] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, May 3-9 2011, pp. 3719–3726.

[15] V. Shapiro and I. Tsukanov, "Implicit functions with guaranteed differential properties," in *ACM symposium on Solid modeling and applications*, ser. SMA '99. New York, NY, USA: ACM, 1999, pp. 258–269.

[16] A. Fuhrmann, G. Sobottka, and C. Gross, "Distance fields for rapid collision detection in physically based modeling," in *International Conference on Computer Graphics and Vision*, 2003.

[17] X. Zhu, H. Ding, and S.-K. Tso, "A pseudodistance function and its application," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 344–352, April 2004.

[18] C. T. Lawrence and A. L. Tits, "A computationally efficient feasible sequential quadratic programming algorithm," *SIAM Journal on Optimization*, vol. 11, pp. 1092–1118, 2001.

[19] F. Kappel and A. V. Kuntsevich, "An implementation of shor's r-algorithm," *Computational Optimization and Applications*, vol. 15, pp. 193–205, 2000.

[20] Y. Nesterov and J. Vial, "Homogeneous analytic center cutting plane methods for convex problems and variational inequalities," *SIAM Journal on Optimization*, vol. 9, no. 3, pp. 707–728, 1999.

[21] 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.

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

[23] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimentional space," *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pp. 193–203, April 1988.

[24] C. J. Ong and E. G. Gilbert, "Fast versions of the gilbert-johnson-keerthi distance algorithm: additional results and comparisons," *IEEE Transactions on Robotics*, vol. 17, no. 4, pp. 531–539, 2001.

[25] ——, "Growth distances: new measures for object separation and penetration," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 888–903, December 1996.

[26] 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.

[27] 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.

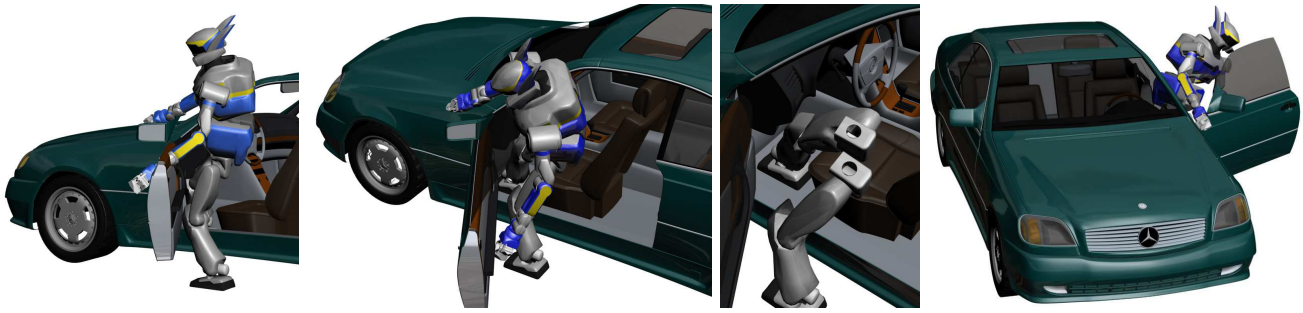


Fig. 20. Enter-car scenario. From left to right: without collision avoidance constraints, successful collision-free posture from three different view points.

[28] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2(4), pp. 505–518, 2005.
[29] 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.
[30] 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.

PLACE PHOTO HERE

**Adrien Escande** received the MS degree in 2005 from École des Mines de Paris, France and the Ph.D. degree in 2008 in robotics from Université d'Évry Val-d'Essonne, France after spending three years in the CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/CRT, Tsukuba, Japan. He then worked as a research scientist in CEA-LIST at Fontenay-aux-Roses, France, until the end of 2012 and is now back at JRL. His current research interests include whole-body planning and control for humanoid robots and mathematical optimization for robotics.

PLACE PHOTO HERE

**Sylvain Miossec** is Assistant Professor at University of Orléans. He obtained his master's degree (2001) and Ph.D. (2004) from École Centrale de Nantes, France. He was then post-doctorate at the CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/CRT with a two-years JSPS fellowship and as a research associate at the Centre National de la Recherche Scientifique (CNRS) for another two years. His research interests include biped robots, humanoid robots, optimal motion, walking control and stability, multi-body simulation, and optimal design of robots.

PLACE PHOTO HERE

**Mehdi Benallegue** is currently a research associate at the LPPA, Collège de France. He obtained PhD degree (2011) from the Université Montpellier 2, France and his master's degree (2008) from the University of Paris 7, France and the ingénieur degree (2007) from the Institut National d'Informatique (INI), Algeria. He conducted his PhD research at the CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/CRT and at the INRIA Grenoble. His research interests include computational geometry, humanoid robots, locomotion, imitation, learning and state observers.

PLACE PHOTO HERE

**Abderrahmane Kheddar** received the ingénieur degree from the Institut National d'Informatique (INI), Algeria, and his master's and PhD degrees in robotics from the University of Paris 6, France. He is currently Directeur de Recherche at CNRS and the Director of the CNRS-AIST Joint Robotic Laboratory, UMI3218/CRT, Tsukuba, Japan. He is also leading the Interactive Digital Human team at CNRS-UM2 LIRMM, Montpellier, France. His research interests include haptics and humanoids. He is a founding member of the IEEE/RAS TC on haptics and the TC on model-based optimization. He was with the editorial board of the IEEE TRANSACTIONS ON HAPTICS (2007-2010). He is presently an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS and the Journal of Intelligent and Robotic Systems. He is a member of the IEEE/RAS.