Fast $\mathcal{C}^1$ Proximity Queries using Support Mapping of Sphere-Torus-Patches Bounding Volumes

Mehdi Benallegue
CNRS–LIRMM, France
AIST/CNRS JRL, Japan

Adrien Escande
AIST/CNRS JRL, Japan

Sylvain Miossec
PRISME-Univ. d'Orléans, France

Abderrahmane Kheddar
CNRS–LIRMM, France
AIST/CNRS JRL, Japan

***Abstract*—STP-BV is a bounding volume made of patches of spheres and toruses. These patches are assembled so that a convex polyhedral hull is bulged, in a tunable way, into a strictly convex form. Strict convexity ensures at least $\mathcal{C}^1$ property of the distance function –and hence, its gradient continuity. STP-BV were introduced in our previous work [1], but proximity distance queries were limited to pairs of STP-BV covered objects. In this work we present an alternative to achieve fast proximity distance queries between a STP-BV object and any other convex shape. This is simply made by proposing a support mapping for STP-BV to be used with GJK algorithm [2]. Implementation and experiments of the proposed method and its performance are demonstrated with potential applications to robotics and computer graphics.**

I. Introduction

The distance function, its properties and its potential application in robotics planning and optimization were nicely studied by the seminal work of Gilbert and Johnson [3]; they also elegantly proved the existence, under certain conditions, of generalized gradients and directional derivatives of the distance function that can be used efficiently in optimized trajectories generation. Another of their result proved –as we motivated differently and independently in [1]– that the gradient continuity of the distance function is guaranteed if one object, from the pair being tested, is strictly convex. In [3], they revealed about achieving such a property by slightly budging objects within a given tolerance margin. However, they did not provide a method to achieve practicly this property. In our previous work [1], we proposed a new method to implement this idea by wrapping sets of 3D points cloud –representing vertices of a graphical geometric model of a given object (e.g. any robot's link)– with a new bounding volume called STP-BV having the following properties:

- it is build from patches of spheres and toruses (STP);
- it is at least $\mathcal{C}^1$;
- it can be seen as a tunable *regularization* of the polyhedral convex hull as it can be continuously morphed from the convex polyhedral hull to the sphere that includes all the object;

In fact our BV can be seen as a merging of two properties which can not be found together in any existing known BV taken alone (see [4] for a list of BVs): it merges the smoothest and simpler know convex BV (spheres), with the best known volume-ratio convex-fit of any object: the polyhedral convex hull (PCH).

Yet, in our previous work [1], a construction method for STP-BV was proposed. Then, we used V-Clip [5] as a basis for distance queries which were restricted only on a pair of STP-BV. This drawback must be discarded since it is sufficient to have only one object as strictly convex to ensure the continuity of the distance's gradient. This is very important because it has a nice consequence in robotics: it is indeed enough to cover any robot with STP-BV to guarantee the gradient continuity of the distance function computed with any other obstacle. Indeed, the continuity of the gradient induces continuity of the low level velocity control in any tracking task which makes use of the proximity distance. Examples of such tasks could be reaching a target with auto-collision and collision avoidance [6] plus the included references, tracking a trajectory with a distance clearance [7], making a contact with the environment by reducing the distance to zero [8]... The space limitation does not allow to discuss many other remarkable work in the field.

This paper proposes a new implementation of STP-BV proximity queries using a hierarchical approach instead of a feature-based approach, yet we keep track of the coherence and spatial properties to speed distance queries. Our novel contribution in this paper solves the following:

- In [1], the distance query was made only between a pair of STP-BV covered objects. This is because we used a feature-based approach in which distance computation are based on properties of the related voronoi region structure. It is easy to draw such properties between pairs of voronoi region representing the same geometrical features (pairs of STP-BV or pairs of PCH) and exploit neighboring region for time and spatial coherence. This generalization would have been extremely difficult to do if each object of a pair has a different model representation and a different topology for the related voronoi regions respectively.
- A consequence of the previous remark is that in [1], the distance computation was restricted to pairs of STP-BV objects and was achieved in two steps: the first step uses the underlying polyhedron convex hull (PCH) –obtained from the STP-BV vertices (the center of the small spheres)– and use V-Clip to find the closest features of a pair of underlying PCH. Once found, the pair underlying PCH features allows knowing the potential closest pair of STP-BV features (small sphere, big sphere or torus) respectively; this is made through heuristic association rules between STP-BV features and the underlying

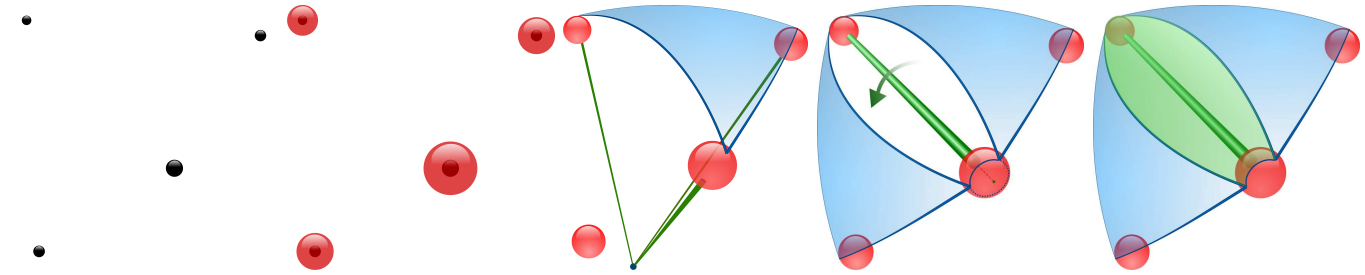

Fig. 1. STP-BV construction steps, from left to right: point clouds (vertices of an object), building vertice-spheres (small radius spheres for security margin), linking vertices-spheres with face-sphere (big radius spheres), closing neighboring face-spheres by edge-toruses (equivalently, the edge-torus is obtained by rotating one face-sphere arc toward the neighboring one).

CPH features. Finally the distance is computed between thre possible cases: two spheres, two toruses or a torus and sphere. First, even if they are very rare, there are cases where (i) the vertices used for STP-BV might not be convex, and (ii) the heuristics association rules might fail. These degenerate cases suggest particular programming efforts for their identification and treatment.

These drawbacks are definitely solved in this work. This is made thanks to a hierarchical approach of the distance computation such as the well known GJK algorithm [2], up-to-date implementation of which allows fast distance queries between any convex shapes, as far as a support function exists for these shapes [9] [10]. We thus propose a support function for STP-BV. We also made several optimizations in the STP-BV and GJK implementation and improved the performance of distance queries as will be seen in benchmark instances. Namely, we also make use of time and spatial coherence when this is allowed.

II. Background: STP-BV

A. Basics

A STP-BV is a bounding volume made of parts of spheres and toruses. The basic idea behind it is to round-off the flat parts of a polyhedron convex hull (edges and faces) from which distance gradient discontinuities are originated. To do so, we perform, in first approximation the following:

- each vertex is replaced by a sphere with small radius r, called *small sphere*,
- each face is associated with a part of *big sphere* with a radius R, that is tangent to the small spheres attached to the vertices of this face,
- each edge is recovered by a part of torus connecting the big spheres of the adjacent faces.

Parts of STP-BV are illustrated on Fig. 1, while a full STP-BV is depicted on Fig. 2. This description is only an approximation because, due to the curvature of the big spheres, some small spheres are not part of the STP-BV while their associated vertices were part of the original PCH. The big spheres define triple of small spheres and thus triple of vertices and faces. The polyhedron associated to the STP-BV might thus be different from the original PCH. We call this new polyhedron the *underlying PCH*.

r is the minimal distance between the original PCH and the STP-BV. It defines a security margin for the collision detection. R controls the curvature of the STP-BV, and thus the regularity of the distance gradient. When $r \to 0$ and $R \to \infty$, the STP-BV tends to the original PCH, Fig. 2.

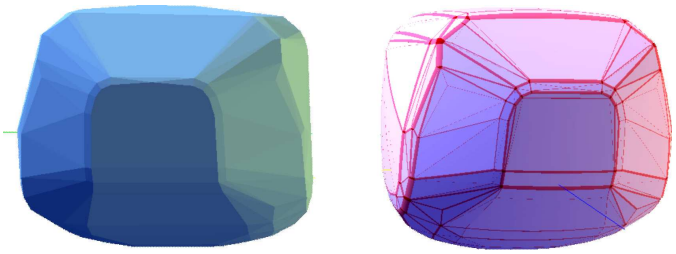

Fig. 2. A robot part (humanoid chest): the polyhedral convex hull bounding volume (PCH-BV), left; and the sphere torus patches bounding volume (STP-BV), right: light pink (big spheres patches), pink (toruses patches), red spots (small sphere patches).

B. Construction

The STP-BV building is similar to the gift wrapping algorithm (or Jarvis march) for PCH, Fig. 1. For given r and R, we start from a cloud of points that have been covered by small spheres. We first look for a triple of small spheres such that there exists a big sphere tangent to them and containing all the other small spheres. The vertexes define three edges. We turn the big sphere around one of them until it becomes tangent to another small sphere. This defines two new edges. Turning the sphere around the edge defines an inner volume that is the part of torus we need. By turning around each new defined edge until we reach edges we already met, we completely wrap the cloud of points and end up with its STP-BV.

III. STP-BV Support Function

Prior to further discussion, we recall quickly the definition of the key concept and step in GJK that is the *support mapping*, which is made through as fast as possible support function.

A. Support function

The *support function* of a compact A is the mapping s_A which associates to each vector $\mathbf{v}$ the point of A such as:

$$\mathbf{v}.s_A(\mathbf{v}) = \max\{\mathbf{v}.\mathbf{a} : \mathbf{a} \in A\}$$

where $s_A(\mathbf{v})$ is the support point of the object in the direction $\mathbf{v}$. In other words, the support mapping gives the farthest point on the object's surface in this given direction.

For convex objects, the support mapping also associates to each vector the point of the object which admits this vector as normal on the surface; for strictly convex objects, this point is unique. The computation of the support function reduces therefore to find the point which admits $\mathbf{v}$ as normal.

Since STP-BV are made from bounded patches of spheres and toruses as primitives (i.e. basic geometric features), we divided in two steps the computation of the support point of an STP-BV in the direction $\mathbf{v}$:

1) find out what primitive (i.e. what STP-BV's patch) contains $\mathbf{v}$ as normal, and then
2) calculate the support mapping of this primitive.

The second step is reduced to use the support function for a sphere or that of a torus; both are very fast. The first step is less trivial; let us define basic data structures that are necessary for the STP-BV's features search. Based on these data structure, several feature search method will be presented.

B. Basics and data structures

1) STP-BV Voronoi Diagram: The voronoi diagram of an STP-BV is a division of the outside space in many cells or regions such as each feature $\mathcal{F}$ of the STP-BV has its cell $C_\mathcal{F}$ so that a point p outside the STP-BV belongs to $C_\mathcal{F}$ iff p is closer to $\mathcal{F}$ than any other STP-BV features.

Several observations can be made at this stage:

- $C_\mathcal{F}$ is the set of points whose projection on the STP-BV is on $\mathcal{F}$;
- the vector $\overrightarrow{p'p}$, p' being the projection of p on the STP-BV, is normal to the STP-BV at p', so that any point $q = p' + k \cdot \overrightarrow{p'p}$, $k \in \mathbb{R}^+$ lies in $C_\mathcal{F}$;
- any Voronoi cell is limited by straight line beams passing by the points of the feature border and following the directions of normal vectors at these points.

2) The Vector Voronoi Diagram: The Vector Voronoi Diagram of an STP-BV is a division of the unit sphere $\mathcal{S}^2$ in cells, such that each feature $\mathcal{F}$ has its cell $\mathcal{C}_\mathcal{F}$ and that $\mathcal{C}_\mathcal{F} = \Gamma(\mathcal{F})$ where Γ is the Gauss map [11]. Thus, a unit vector $\mathbf{v}$ belongs to the $\mathcal{C}_\mathcal{F}$ iff there is a point on $\mathcal{F}$ such that the unit normal vector to the STP-BV at this point is equal to $\mathbf{v}$.

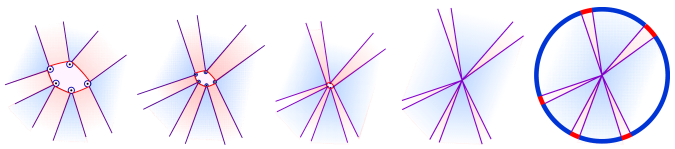

Fig. 3. Obtaining vectorial Voronoi diagram.

Intuitively, this is similar to shrinking the STP-BV to a point while augmented with its voronoi diagram primitives, see Fig 3. During this transformation, the voronoi diagram varies continuously. When the STP-BV reduces to the single point, the limit of the voronoi diagram is a decomposition of the vector space. We may consider, without loss of generality the point equals to $\vec{0}$; $p \in C_\mathcal{F}$ iff $\overrightarrow{p'p}$ is normal to the STP-BV at a point on $\mathcal{F}$. The vector voronoi diagram is the intersection of this limit diagram with the unit sphere and the naming *Vector Voronoi Diagram* is the simple interpretation of the fact that the partition of the sphere is simply the voronoi diagram, but reduced to the vector space.

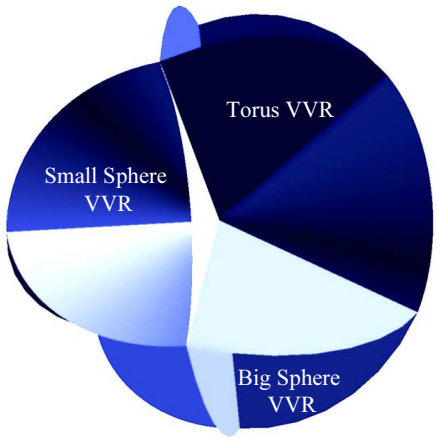

Fig. 4. All Vector Voronoi Regions (VVR) for a regular STP-BV covered tetrahedron.

3) Vector Voronoi Region: The Vector Voronoi Regions (or Cells), noted VVR, of STP-BV are such that, Fig 4:

- The small sphere VVR is limited by a finite set of part of cones whose apex is the origin and whose axes are respectively the adjacent edges, Fig 5.
- The big sphere VVR is limited by three planes passing by the origin.
- The torus VVR is bounded by two cones and two planes.

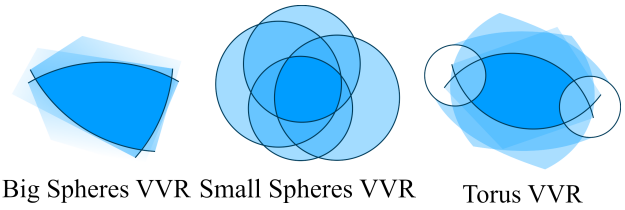

Fig. 5. Steregraphic projection of vectorial voronoi regions.

A stereographic illustration (2D projection) of the vector voronoi regions is illustrated by Fig. 5 for clarity.

C. Feature search methods

The feature search step reduces to finding which VVR contains a given direction vector $\mathbf{v}$. We have seen previously that the VVR of STP-BV's features are limited by a finite set of cones and planes. We can check if a given vector is within a VVR simply by testing if the vector lies in the right side of each boundary. This is made simply using a dot product between the vector and the axis of the cone or the plane's normal (we consider the plane as a particular cone having

$\pi/2$ angle). Indeed, for a cone c with normalized axis $\mathbf{a}$ and angle θ, we have :

$$\mathbf{v} \text{ inside } c \Longleftrightarrow \frac{\mathbf{v}.\mathbf{a}}{\parallel \mathbf{v} \parallel} > \cos\theta$$

This evaluation allows also having an evaluation of the distance between the vector and the violated boundary. We noticed that a VVR may be expressed exclusively with its boundaries, and each boundary simply with its axis and its angle cosine.

Now that we have a method to check if a vector lies inside a VVR, it remains to find to which VVR a given vector belongs to. We propose hereafter three non-exclusive methods which resolve this.

1) Naive method: It consists simply in checking for each VVR if the vector is inside or not. This technique always finds the feature in demand. The complexity of this method is however $\mathcal{O}(n)$, n being the number of features in the STP-BV, and it cannot be easily used to take advantage of time and spatial coherences. A possible method would be a spiral march around the last feature found, which is really difficult in such irregular patches.

This naive method is certainly not optimal, but since it is very robust, we used it to check the results of the following optimized ones.

2) Guided march method: This method is inspired by the hill climbing-based polyhedron support function. When we know that a vector does not belong to a given VVR, we also know to which boundary the vector is on the wrong side. Thus it is natural to check if the vector belongs to the VVR of the feature lying in the other side of this boundary. The problem is when many boundaries are violated; the choice of the neighbor must not be arbitrary, because infinite loops may occur. The experience has shown that the following choice always reaches the right VVR whatever the beginning feature:

- *Big spheres*: choose any neighbor of the good side of a boundary.
- *Torus*: go prior to big spheres nearby if one of them is in the right side of its limit plan. Otherwise, go to small sphere.
- *Small sphere*: take the torus nearby whose common limit with the small sphere is the farthest from the given vector.

The infinite loop case, although it has never been met in real situation, could be easily detected after n tests, n being the number of features of the STP-BV, then the algorithm would behave like the naive method that always finds the right feature.

There is no theoretical complexity calculation about this method, but we can evaluate it to $\mathcal{O}(\sqrt{n})$, because if the algorithm behaves like expected, it is a march guided by a potential field in a two-dimensional manifold divided into cells.

3) The underlying polyhedron convex hull method: It is an acceleration technique to the guided march method. It works in two steps; the first is to find the support point for the underlying polyhedron convex hull, that can be considered to be always a vertex on the polyhedron. The next step is to start the guided march method from the small sphere corresponding to the support vertex.

The idea of this method is that the support point in the STP-BV for a given direction can not be far from the farthest small sphere in this direction. We know also that the farthest vertex according to a given direction corresponds exactly to the farthest small sphere, so if we find this vertex, we can be sure to start the guided march from the closest small sphere. Hence, the underlying polyhedron structure allows us to accelerate the farthest point search. However, we can not use directly the STP-BV underlying polyhedron (i.e. the polyhedron having the center of the small spheres as vertexes which are linked by edges by keeping torus patches' topology); because there are some extreme cases where it is not convex. Thus we have to use the underlying polyhedron convex hull recomputed from the small sphere centers, and use it to find the farthest vertex.

The VVR of the small sphere is always strictly contained in the VVR of the corresponding vertex of the underlying polyhedron convex hull. It means that the second step, starting from a small sphere, never reaches another one, and it can find the searched feature at distance less than 4 from the initial small sphere.

The complexity of this method depends on the complexity of the first step, as the second one has constant time. There exists a $\mathcal{O}(\log n)$ complexity polyhedron support function computation method (see a thorough discussion in [10]) but it does not take profit from the time-space coherence. Thus we use the Hill climbing method which is easily adapted to keep the latest support data.

D. Support functions for STP-BV features

Any previously chosen feature search method returns a sphere (big or small) or a torus.

The support function of a sphere $\mathcal{S}_{c,l}$ (with a center c and a radius l) is known and simply computes in:

$$s_{\mathcal{S}_{c,l}}(\mathbf{v}) = c + \frac{l.\mathbf{v}}{\|\mathbf{v}\|}$$

For the case of torus features, we must keep in mind that the word "torus" is a misnomer. Indeed, we consider in the torus the inter-penetrated part [1]. We can compute the support function of this entire portion, but we know that the belonging test automatically excludes the peaks. The torus portion is the result of a big sphere rotation around an edge, this makes the center of the big sphere form a circle C. The support point of the considered part for a vector $\mathbf{v}$ is given by the sum of the support point of the circle C according to the vector $-\mathbf{v}$, and the support point of the big sphere with radius R and centered in the origin O according to the direction $\mathbf{v}$; that is:

$$s_A(\mathbf{v}) = s_C(-\mathbf{v}) + s_{\mathcal{S}_{O,R}}(v) = s_{\mathcal{S}_{s_C(-\mathbf{v}),R}}(v)$$

IV. Implementation and Performances

Previous theoretical results have been implemented in C++ into a packaged library called SPQ for Smooth Proximity Queries. It is has been implemented as a standalone code so

that it can be integrated to several applications, but it has been optimized to run in real-time since our primary focus is to use it in the low-level control of humanoid robots. Data structure is also enhanced to contain additional information for exploiting space and time coherencies (e.g. last visited feature, last distance witness points, etc.). The library is modular enough to be extended to any other convex geometry [9] and BVs such as OBB, AABB, Spheres, STP-BV, PCH-BV, and superellipsoids (SE-BV).

A. Performance study

At first, performances of the support function methods have been tested separately. All the tests have been run on a Pentium 4 PC 3.8GHz and 2GO RAM. We chose a complex object of 1100 vertexes (that represent a link of the HRP-2 humanoid robot) as bench; in order to have a variable number of features, we simply changed the ray of the big sphere of the STP-VB. These tests allow to confirm that the complexity of the support function computation is sub-linear.

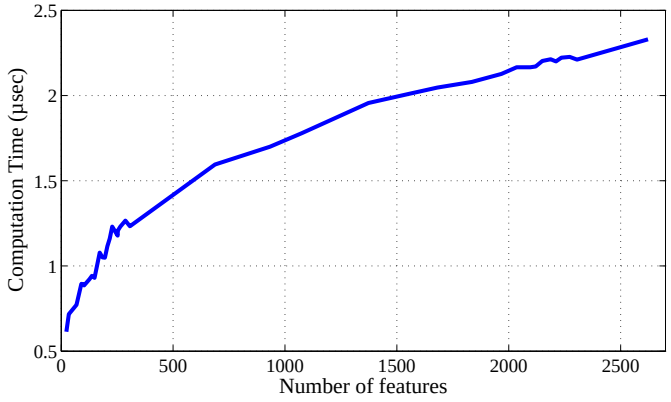

Fig. 6. Performance for the guided march based function support computation.

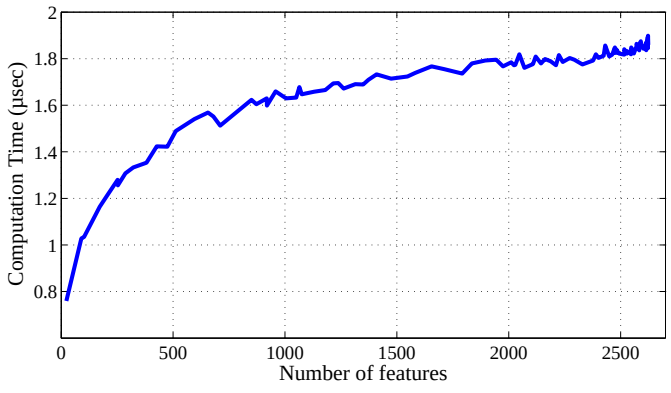

Fig. 7. Performance for the underlying polyhedron convex hull based function support computation.

The Fig. 6 illustrates the obtained performance of the *guided march method* which allows to confirm its $\mathcal{O}(\sqrt{n})$ complexity. Fig. 7 illustrated obtained results of the *underlying polyhedron convex hull* method which allows to confirm its $\mathcal{O}(\sqrt{n})$ complexity with a better multiplier coefficient.

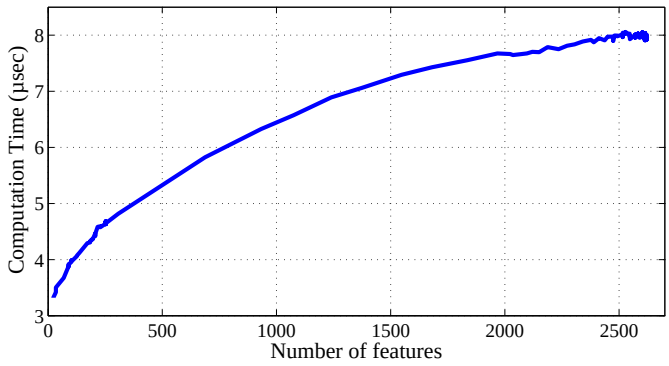

Fig. 8. Performance for a complete GJK proximity queries between similar pairs of STP-BV.

In the following we undergo the performance evaluation of the complete proximity query. We used an object having 300 vertexes, which is also an HRP-2 link. Fig. 8 shows obtained performances of the overall query without spatial and temporal coherence (i.e. the worst case) and using the guided march method for computing the support function of the STP-BV. The precision asked for the query is 10^{-6} (which can be quoted in meters for our case studies, clearly this is far too much from what one requires to avoid collision in robotics!). Again the overall complexity is sub-linear relatively to the complexity of the object.

B. Comparative study

Table I relates obtained performance results in terms of number of proximity queries for different convex volume bounded objects. This object has been covered with three different bounding volumes: the polyhedral convex hull (PCH), our proposed STP-BV, and a best fit convex superellipsoid (CSE-BV). The support function for the SE is computed analytically[1] and not obtained from its polyhedral approximation. Tests have been performed on randomly positioned and oriented pairs of the same object.

TABLE I
NUMBER OF PAIR PROXIMITY QUERIES PER SECOND.

	PCH-BV	STP-BV	CSE-BV
PCH-BV	795123		
STP-BV	584567	525394	
CSE-BV	319216	286368	187429

The results show that faster queries can be obtained using pairs of PCH-BV but without guarantee of distance gradient continuity. For this object, it appears that STP-BV queries would require 39% more time then PCH-BV whereas CSE-BV queries would require 145% more time than PCH-BV. But this is only a tendency that is likely to change depending on

[1]This is the object of another paper.

the complexity of the object and the application requirements. In another hand, there is actually a benefit in using a double STP-PCH BV representation for each robot's link. The reason why this is interesting appears clearly for humanoid auto-collision avoidance. For a distance query between a given pair of bodies, it is better to choose an STP-BV representation for the body containing less STP-BV features, and a PCH-BV representation for the other. This reasoning certainly extends to any pair of bodies, i.e. including the objects of the environment. This also means that both BV need to be stored in memory to use the appropriate one at will.

V. Application Examples

Our method is packaged into a `C++` code and integrated to a humanoid avatar interactive simulator framework we have developed. It can also be used as a standalone code to be integrated in any other application. The Fig. 9, is a screen snapshot of the video attached to the paper. In this simulation, all the 31 links of the HRP-2 humanoid robot are randomly moving and proximity distances are computed between all pairs of objects.

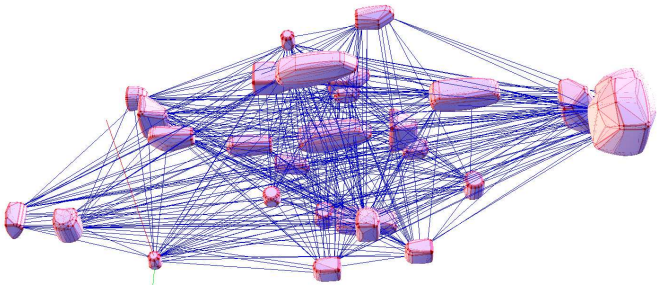

Fig. 9. STP-BV covered HRP-2 links, distance query is performed between all pairs of links.

For the example of the Humanoid HRP-2, we are able to compute necessary pairs for auto-collision (here we took the worst case) in less then 0.7msec of time. When temporal and spatial coherences are taken into account, this time reduces to 0.5msec for the overall robot. This means that we are able to integrate auto-collision into the low-level control.

The Fig. 10 illustrates pairs of distance proximity queries for a virtual avatar, here, the obtained STP-BV are rendered in transparency to see the under-covered avatar. The distance is illustrated by the (red) line linking computed witness points for each selected pairs of body.

VI. Conclusion

We present a substantial improvement of our STP bounding volume [1] which allows fast $\mathcal{C}^1$ proximity queries between convex shapes. Provided that (i) any object can potentially be decomposed into a collection of convex sub-objects, and (ii) it suffices that only one object be strictly convex to ensure continuous gradient of the distance function [3], this method has certainly potential use in robotics, humanoids and virtual avatar whole-body free-collision motion generation control or

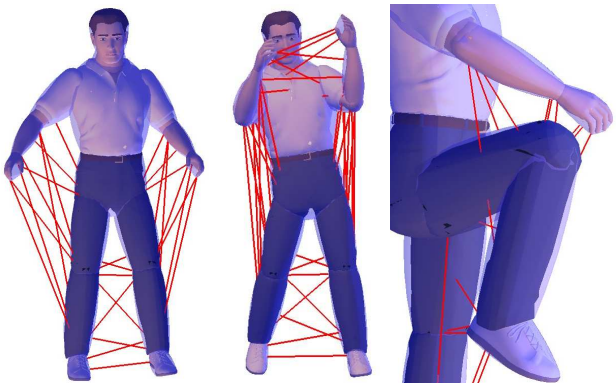

Fig. 10. Auto-collision avoidance with a virtual avatar.

optimization. For the time being, we proved that the obtained performance are satisfactory and the implementation is robust. As future work, this software will be further code-optimized and integrated in a low level control of a real humanoid robot as a continuation of [6].

Acknowledgment

This work is partially supported by grants from the ImmerSence EU CEC project, Contract No. 27141 (FET-Presence) under FP6 www.immersence.info.

References

[1] 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.

[2] 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.

[3] 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.

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

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

[6] 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*, Pasadena, California, 19-23 May 2008, pp. 3200–3205.

[7] V. Patoglu and R. B. Gillespie, "Feedback-stabilized minimum distance maintenance for convex parametric surfaces," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 1009–1016, October 2005.

[8] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning support contact-points for acyclic motion and experiments on HRP-2," in *International Symposium of Experimental Robotics*, Athens, Greece, 14-17 July 2008.

[9] E. Gilbert and C.-P. Foo, "Computing the distance between general convex object in three-dimentional space," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 53–61, February 1990.

[10] 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.

[11] T. Banchoff, T. Gaffney, and C. McCrory, "Cusps of the gauss map," *Research Notes in Mathematics*, no. 55, 1982. [Online]. Available: http://www.math.brown.edu/~dan/cgm/index.html