A distance bound for nonconvex polyhedral models in close proximity

P. Jiménez C. Torras

Institut de Robòtica i Informàtica Industrial (CSIC - UPC)

Llorens i Artigas 4-6, E-08028 Barcelona, Spain

e-mail: pjimenez@iri.upc.es, ctorras@iri.upc.es

Abstract

In many applications, it suffices to know a lower bound on the distance between objects, instead of the exact distance itself, which may be more difficult to compute. Such an easy-to-compute lower bound on the distance between two nonconvex polyhedra is presented here, which doesn't require a decomposition of the original polyhedra into convex entities. Furthermore, a suitable preprocessing of the polyhedra permits lowering the effort needed to compute this lower bound, and improves its quality. Experimental evidence is presented of the promise of this approach for assembly applications.

1 Introduction

Distance computation is a basic issue in many mechanical simulation and computer animation applications. For example, knowing the distance and the relative velocities and accelerations of objects allows to predict the instant where collision may take place between these objects. Exact and approximate distance computation belong, together with collision detection and tolerance verification, to the so called proximity queries (see[1–3] for surveys on the subject).

Very efficient algorithms exist for computing and updating the distance between two convex polyhedra.[4–11] If the polyhedra are nonconvex, these algorithms can only be applied after a preprocessing step, consisting of a decomposition of the polyhedra into smaller convex ones. The algorithm has to compute then the distance between every pair resulting from this decomposition, and its minimum provides the distance between the original polyhedra. Such decomposition introduces many ficticious features which intervene in the distance computations.

Recently, also very fast algorithms have been devised for distance computation between nonconvex polyhedra with triangular faces.[12–15] Again, if the faces of the original polyhedra are polygons of arbitrary complexity, they have to be triangulated, which can be a costly operation. Triangulation constitutes obviously a preprocessing operation, and therefore has to be performed only once, but the fictitious edges introduced by it have to be considered each time the distance is computed.

In sum, computing the exact distance between two objects may be costly if the complexity of the objects is high. However, it often suffices to compute a lower bound which,

together with bounds on the velocities, permits to set time bounds on collision instants, which is enough in most collision detection contexts. This strategy, of course, makes only sense if the lower bound is easier to compute than the distance itself. If the objects are distant, such lower bound can be easily computed from simple enclosing volumes (boxes, spheres, ellipsoids –like in,[16] or other). Tighter bounds on the distance can be obtained by organizing such simple enclosing volumes in hierarchies that describe the object with a progressive degree of accuracy (the root bounds the whole object, whereas the leaves correspond to elemental features like triangles or polygonal faces in the case of polyhedra). This allows to compute a distance bound up to a given tolerance (which may be dependent on the available time for performing computations) or even to direct the search towards the closest feature pair. Of course, major benefits are obtained from the situations where the search can be interrupted in the first levels of the hierarchies (because the corresponding tolerance is enough for the purposes) or if significant portions of the hierarchies can be pruned away (because it can be decided that they cannot contain the solution). This approach has inspired a large number of works.[17–22] However, the efficiency of this approach may be questionable if the objects are in close proximity, as may be the case of assembly operations when the workpieces are close to their fitting (matching) positions.

In this paper, we present a theoretical curiosity that shows promise of having practical interest. Extending the seminal ideas of Canny[23] and Donald[24] on interference detection predicates, Thomas and Torras[25] developed an interference test for nonconvex polyhedra which doesn't require decomposition into convex entities. Here we show that, by replacing predicates by their corresponding continuous functions (as suggested in[25]), a lower bound on the distance between the polyhedra is readily obtained. These ideas are displayed and

discussed in Section 2, where a correctness proof is provided. The good news are that these basic functions are simply computed from vertex coordinates, face normals, and edge director vectors, and no auxiliary geometric constructs are needed.

The lower bound is obtained by computing the value of the function for all possible pairings between the edges of one polyhedron and the faces of the other one, which means quadratic complexity for the algorithm. It is possible to lower this computational effort by discarding some edge-face pairs on the basis of their relative orientations, as shown in Section 3. Experiments reported in Section 3.3 show that this pruning does not only avoid unnecessary computations, but also enhances the quality of the lower bound.

It should be noted that the main contributions of this work are the proposal of a lower distance bound computed from basic contact functions, which is valid for nonconvex faces, as well as an orientation-based pruning strategy, complementary to the volume-based approaches cited above. Actually, the two approaches (volume-based and orientation-based) can be combined in order to exploit their respective pruning capabilities, as was done in.[13] This issue is discussed in Section 4.

Finally, some conclusions are drawn in Section 5.

2 Determining a lower bound on the distance between two general polyhedra

2.1 Basic functions

When determining the distance between two polyhedra, the immediate and most natural approach consists in computing and comparing the distances between their boundary features (vertices, edges, and faces). Vertex - vertex distances are simple point-to-point distances, but in all other possible pairings a decision process is necessary to determine which kind of elemental distance formula has to be applied: an edge-face distance, for example, can be a point-to-point, a point-to-plane, or a line-to-line distance.

In our approach, we need to compute only two types of basic distance functions, which we call type-A and type-B functions, following the well-established nomenclature of basic contacts.[23]

A **type-A function**[1], associated with a vertex v of one polyhedron and a face f of the other one, is a signed point-to-plane distance from v to the plane containing f:

$$A_{v,f} = \langle f, v - v_i, \rangle$$

where, in abuse of notation, vertices stand for their coordinates vectors and faces for their normals, and v_i is any vertex of f.

A **type-B function**, associated to edge e_m of one polyhedron and edge e_n of the other

[1] In type-A and type-B functions, vertices, edges and faces are variables dependent on the relative pose of the involved polyhedra.

one, is a signed line-to-line distance between the supporting lines of e_m and e_n:

$$B_{e_m,e_n} = \langle e_m \times e_n, v_l - v_k, \rangle$$

where the edges stand for their director vectors, $v_k = \partial^+ e_m$ and $v_l = \partial^- e_n$ (∂^+ and ∂^- are the halfboundary operators – in this case, they refer to the endpoints of the edges).

These two basic functions are the elemental constituents of our lower distance bound, as shown next.

2.2 The lower distance bound function

It was suggested in[25] that the continuous version of the interference detection predicate proposed therein, and experimentally validated in,[26] could lead to a lower bound on the distance between two polyhedra. This entails replacing basic predicates by basic functions, AND by *min*, OR by *max* and XOR by *smin* (signed minimum), leading to the following composite function that has to be applied to every edge-face pair of two polyhedra:

$$\begin{aligned} D_{e,f} = min(\quad & smin(A_{\partial^+ e,f}, A_{\partial^- e,f}), \\ & smin_{e_f \in \partial f}(min(smin(A_{\partial^+ e_f,f_e}, A_{\partial^- e_f,f_e}), smin(A_{\partial^- e_f,f_e}, B_{e,e_f})))), \end{aligned} \tag{1}$$

where f_e stands for an arbitrary plane containing e, and ∂f is the set of edges bounding f. The *max* and *min* operators have their usual meaning, whereas *smin* returns the lowest absolute value of the operands, and its sign is positive if an odd number of operands is positive, negative otherwise.

To ease notation, we will call:

- $a_{e,f} = smin(A_{\partial^+ e,f}, A_{\partial^- e,f})$

- $c_{e_f} = min(smin(A_{\partial^+ e_f,f_e}, A_{\partial^- e_f,f_e}), smin(A_{\partial^- e_f,f_e}, B_{e,e_f}))$

- $b_{e,f} = smin_{e_f \in \partial f}\{c_{e_f}\}$

Thus, $D_{e,f} = min(a_{e,f}, b_{e,f})$.

Intuitively, $-a_{e,f}$ is a lower bound on the distance from e to the plane supporting f, and $-b_{e,f}$ is a lower bound on the distance between f and the line supporting e.

A lower bound on the distance (ldb) between two polyhedra P and Q is obtained by evaluating $D_{e,f}$ for all possible pairings between the edges of one polyhedron and the faces of the other one:

$$ldb = -max(D_{e,f}), \forall (e,f) \text{ such that either } e \in P \text{ and } f \in Q, \text{ or } e \in Q \text{ and } f \in P. \quad (2)$$

In,[25] it was argued that a lower bound on the distance was obtained due to the fact that type-A functions provide vertex-plane distances, which are lower bounds on vertex-face distances (from $\partial^+ e$ and $\partial^- e$ to f) and on vertex-line distances (from $\partial^+ e_f$ and $\partial^- e_f$ to the line supporting e); and type-B functions provide line-line distances, that is, lower bounds on the line-edge distances (from the line l_e supporting e to edges e_f). There it was also explained that the quality of the ldb could be improved by choosing f_e (which is needed to evaluate $b_{e,f}$) to be a plane that maximizes the distance to the nearest vertex of face f (instead of an arbitrary plane).

A formal proof of the correctness of the lower distance bound (2) is given in the next section. The mentioned plane can be computed efficiently with a divide-and-conquer

algorithm that solves the equivalent 2D problem obtained by projecting the vertices of the face on a plane perpendicular to the edge. Details on this algorithm are given in.[27] However, in practice an easy-to-compute plane, as the one containing the edge and the origin of coordinates, works as well, provided there is some mechanism to avoid degenerate situations, as shown in the experiments.

2.3 Correctness

If there is no interference between the considered polyhedra, $D_{e,f} \leq 0$ for all edge-face pairs, and thus $ldb \geq 0$. Moreover, $ldb = min\{|D_{e,f}|\}$. On the other hand, the minimum distance between the two polyhedra $d_{P,Q}$ can be expressed as the minimum of all the distances between edges and faces of the two polyhedra, $d_{P,Q} = \min\{d(e,f)\}$. Therefore, it is necessary to prove that $D_{e,f} \geq -d(e,f)$, i.e., $|D_{e,f}| \leq d(e,f)$ for all edge-face pairs.

As $D_{e,f} = min(a_{e,f}, b_{e,f})$, two situations have to be considered:

- Suppose $D_{e,f} = a_{e,f}$. It follows that $a_{e,f} \leq 0$ (as $D_{e,f} \leq 0$), i.e., edge e does not intersect the plane supporting face f. The basic functions in $a_{e,f}$ are vertex-plane distances (from the endpoints of e to the plane supporting f), and that of minimum norm is, in this case, a lower bound on the edge-face distance.

- If $D_{e,f} = b_{e,f}$, then $b_{e,f} \leq 0$. This means that the line l_e supporting e does not intersect f. In this case, $d(e,f) \geq d(l_e,f) = min_{e_f \in \partial f}\{d(l_e,e_f)\}$. Thus, it must be proven that $|b_{e,f}| \leq min_{e_f \in \partial f}\{d(l_e,e_f)\}$. Since $|b_{e,f}| = min_{e_f \in \partial f}\{|c_{e_f}|\}$, this reduces to proving that $|c_{e_f}| \leq d(l_e,e_f)$, for all e_f in the boundary of f.

  For any one of these boundary edges,

$$|c_{e_f}| = |min(smin(A_{\partial^+ e_f, f_e}, A_{\partial^- e_f, f_e}), smin(A_{\partial^- e_f, f_e}, B_{e,e_f}))| = |min(smin_1, smin_2)|.$$

Consider the possible combinations of signs of the *smin* functions:

(i) $smin_1 < 0$, $smin_2 < 0$. Then $|c_{e_f}| = max(|smin_1|, |smin_2|)$. The two alternatives are studied below.

(ii) $smin_1 < 0$, $smin_2 > 0$. Now $|c_{e_f}| = |smin_1| = min(|A_{\partial^+ e_f, f_e}|, |A_{\partial^- e_f, f_e}|)$.

(iii) $smin_1 > 0$, $smin_2 < 0$. In this case, $|c_{e_f}| = |smin_2| = min(|A_{\partial^- e_f, f_e}|, |B_{e,e_f}|)$.

(iv) $smin_1 > 0$, $smin_2 > 0$. Here, $|c_{e_f}| = min(|A_{\partial^+ e_f, f_e}|, |A_{\partial^- e_f, f_e}|, |B_{e,e_f}|)$.

In case (i) with $|smin_2| > |smin_1|$, as well as in cases (iii) and (iv), it holds that $|c_{e_f}| \leq |B_{e,e_f}| = d(l_e, l_{e_f}) \leq d(l_e, e_f)$.

In the remaining cases, $smin(A_{\partial^+ e_f, f_e}, A_{\partial^- e_f, f_e}) < 0$, which means that the boundary edge e_f does not intersect the plane f_e. This implies that the minimum distance between e_f and f_e is realized at one of the endpoints of e_f. This vertex - plane distance is a lower bound on the edge - plane distance, i.e., $min(|A_{\partial^+ e_f, f_e}|, |A_{\partial^- e_f, f_e}|) \leq d(f_e, e_f) \leq d(l_e, e_f)$, which completes the proof.

2.4 The degenerate case

Sometimes *ldb* is not a tight lower bound. Particularly, the case where it reports contact, i.e. it is equal to zero, although the polyhedra are apart, is very undesireable. When can this happen?

Suppose both polyhedra are separated and $D_{e,f} = \min\{a_{e,f}, b_{e,f}\} = 0$. Two cases have to be considered:

- $a_{e,f} = 0$ and $b_{e,f} > 0$, which corresponds to a real contact situation (Figure 1).

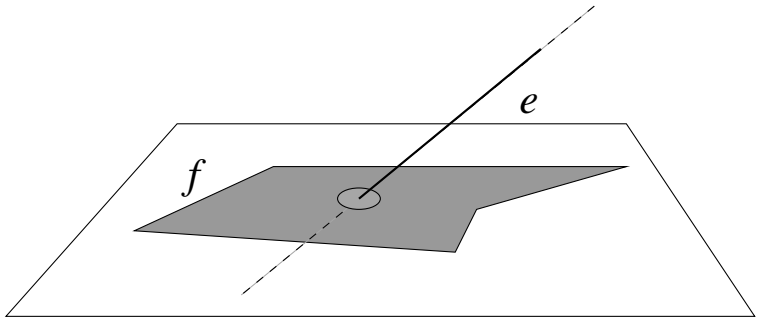

Figure 1: $a_{e,f} = 0$ *and* $b_{e,f} > 0$ *implies a real contact.*

- $b_{e,f} = 0$ and $a_{e,f} \geq 0$. This means that $\exists e_f$ s.t. $c_{e_f} = 0$. We will distinguish two main cases where this can happen, depending on the value of B_{e,e_f}:

  $|B_{e,e_f}| > 0$ Necessarily either $A_{\partial^+ e_f, f_e} = 0$ or $A_{\partial^- e_f, f_e} = 0$ (not both simultaneously, as this would mean coplanarity of e_f and e, which is the other case). In other words, the arbitrary plane containing edge e does also contain one of the endpoints of e_f. Any other plane f_e can be chosen that avoids this circumstance (Figure 2).

  $B_{e,e_f} = 0$ Edges e and e_f are contained in the same plane. Consider the value of $a_{e,f}$:

  $a_{e,f} > 0$ Corresponds to a real contact (Figure 3).

  $a_{e,f} = 0$ There are two subcases:

  - Either $A_{\partial^+ e, f_e} = 0$ or $A_{\partial^- e, f_e} = 0$, but not both. This corresponds to a real contact situation (Figure 4a).

– $A_{\partial^+ e,f} = A_{\partial^- e,f} = 0$ can either correspond to a real contact, as shown in Figure 4b1, or we are facing the only degenerate case, where $ldb = 0$ while no real contact exists (Figure 4b2).

Summarizing, the only situation where $ldb = 0$ and no real contact is happening is the case where e is coplanar to f and its supporting line cuts an edge $e_f \in \partial f$ at any point. When this case is spotted ($A_{\partial^+ e,f} = A_{\partial^- e,f} = B_{e,e_f} = 0$), an alternative, 2D version of function (1) can be used to get a tighter lower bound:

$$D^{2D}_{e,f} = -max_{e_f \in \partial f}(min(smin(A^{2D}_{\partial^+ e,e_f}, A^{2D}_{\partial^- e,e_f}), smin(A^{2D}_{\partial^+ e_f,e}, A^{2D}_{\partial^- e_f,e}))), \qquad (3)$$

where $A^{2D}_{v,e}$ is a signed distance in the plane from vertex v to the supporting line of e. If the sign reported by this function is negative, edge e actually intersects face f.

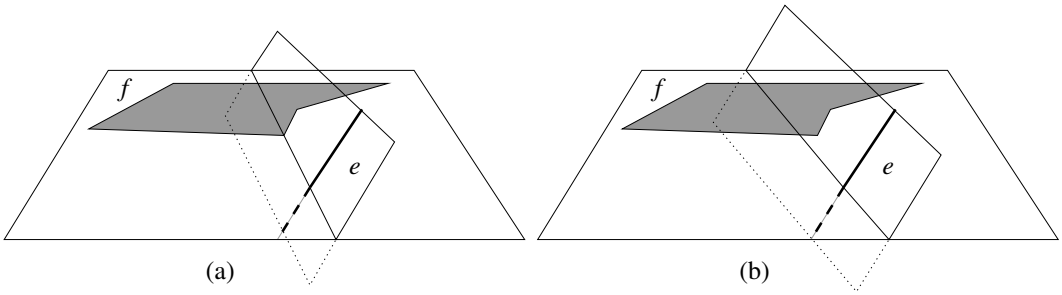

Figure 2: *(a) A bad choice of plane f_e contains one of the endpoints of an edge e_f of 'f'. (b) A different choice of this plane avoids this problem. The edge 'e' appears as a continuous line over or on the supporting plane of 'f', and discontinuous under this plane.*

3 Lowering the computational effort

The search space attached to the evaluation of function (1) applied to all possible edge-face pairings of two polyhedra can be represented as a tree, whose structure is quite similar

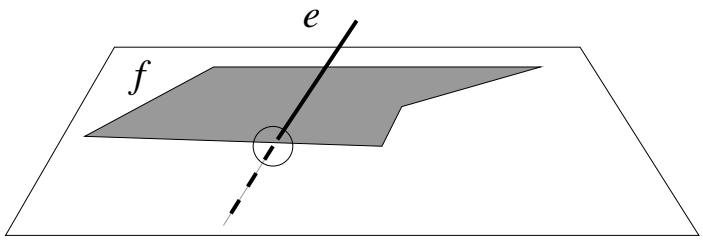

Figure 3: *A real contact (highlighted with a circle) exists such that* $B_{e,e_f} = 0$ *and* $a_{e,f} > 0$.

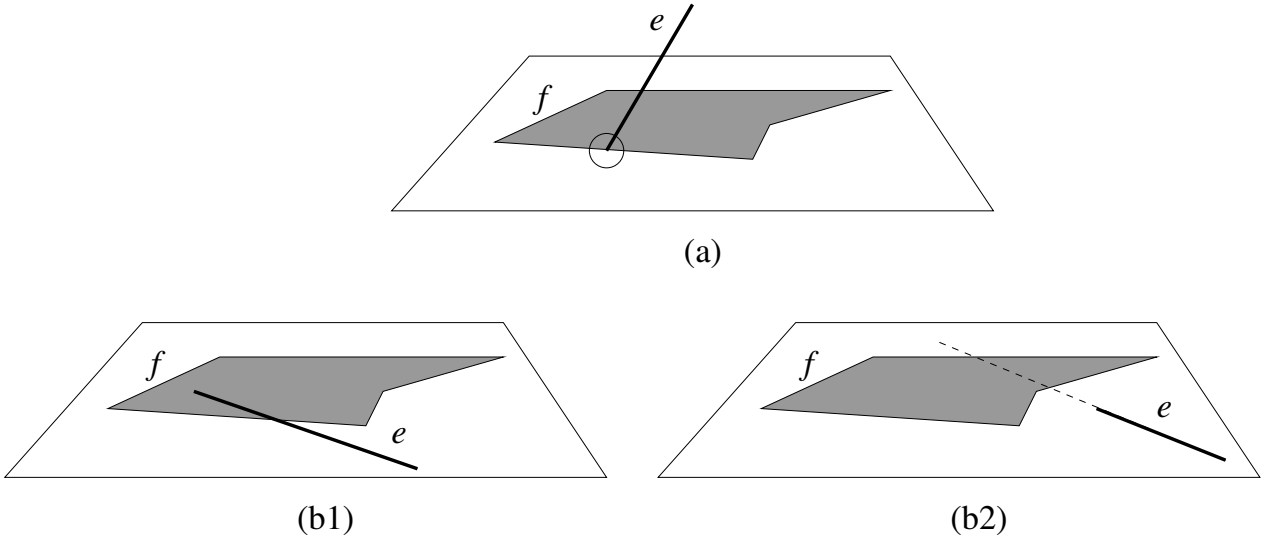

Figure 4: $B_{e,e_f} = 0$ *and* $a_{e,f} = 0$. *(a) Only one of the endpoints of 'e' touches the plane 'f'. Then, necessarily a real contact exists. (b) Both endpoints of 'e' are on 'f', i.e., 'e' and 'f' are coplanar.* $ldb = 0$ *reports either a real contact (b1), or corresponds to the degenerate situation (b2).*

to a MIN - MAX tree. Thus, the well-known α-β pruning strategy can be adapted to cope with the particular structure of the MIN-MAX-SMIN tree. A detailed description, as well as examples and experimental results, can be found in.[27]

However, α-β pruning does not avoid having to compute at least two functions per edge-face pair (in the best case, the computation of $b_{e,f}$ can be skipped). It would be desirable to avoid doing any computation at all for edge-face pairs that cannot possibly realize the minimum distance. This can be achieved after a suitable preprocessing step, as shown next.

3.1 Orientation-based pruning based on applicable contacts

Orientation-based pruning is a preprocessing which consists in eliminating a whole set of edge-face pairings that cannot realize the minimum distance between the polyhedra. This pruning is based on the so-called *applicable* contacts, and has been used previously by the authors in the interference detection context.[26,28] Applicability refers to which features of two translating polyhedra can be brought in contact. As we will see in the next subsection, the minimum distance can only be realized by features whose contact is applicable. Thus, only edge-face pairings involved in applicable contacts need to be considered for computing function (2).

Every contact between the features of two polyhedra can be reduced to a basic vertex-face or edge-edge contact. The two basic contacts are displayed in Figure 5, and the corresponding *applicability conditions* allow one to determine which of these contacts are applicable:[24]

Type A applicability For a given relative orientation between two polyhedra, the contact between a vertex v of one polyhedron and a face f of another polyhedron is applicable iff $\forall v_i$ adjacent to v, $\langle v_i, f\rangle - \langle v, f\rangle \geq 0$.

Type B applicability For a given relative orientation between two polyhedra, the contact between an edge e_m of one polyhedron and an edge e_n of another polyhedron is applicable iff $k_a \neq k_b$, where $k_a = \text{sign}(\langle T_1, f_p\rangle) = \text{sign}(\langle T_2, f_p\rangle)$, and $k_b = \text{sign}(\langle T_3, f_p\rangle) = \text{sign}(\langle T_4, f_p\rangle)$, with $T_i = s_i \cdot (f_i \times e_m), f_i$ adjacent to $e_m, T_j = s_j \cdot (f_j \times e_n), f_j$ adjacent to $e_n; s_i, s_j \in \{+1, -1\}$ such that T_l is oriented towards the interior of face f_l (see Figure 5), and $f_p = e_m \times e_n$ (or the opposite direction, the choice is arbitrary).

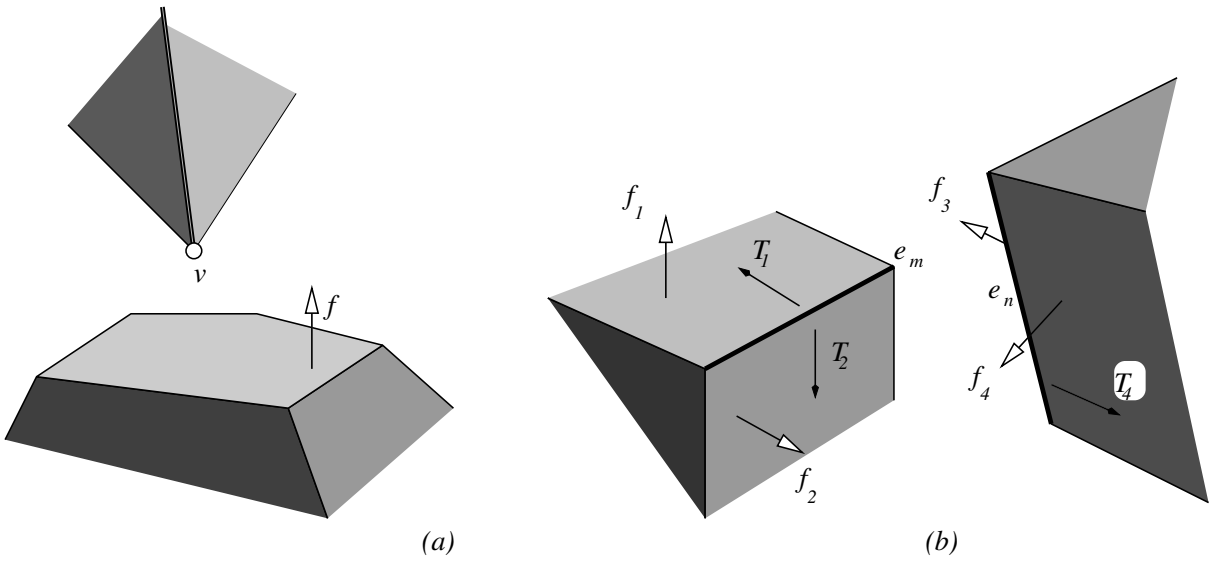

Figure 5: *(a) Applicable vertex - face pairing. (b) Applicable edge - edge pairing.*

Note the correspondence of these basic applicable contacts with the basic distance functions in Section 2.1.

Our preprocessing[2] proceeds in two steps. In the first one, all basic applicable contacts

[2]The representation and algorithms developed to perform this orientation-based pruning efficiently

are determined. In the second step, edge-face pairs related to these contacts are computed: for a vertex-face contact, any one of the adjacent edges to the vertex and the face itself are chosen, whereas for an edge-edge contact the choice falls on any one of the edges and any one of the two faces that are adjacent to the other edge. These chosen edge-face pairs provide a lower bound on the distance if function (2) is applied to them, as proven in the next subsection. Moreover, the experiments show that the quality of the lower bound improves in general with the preprocessing, i.e., gets closer to the actual shortest distance.

The applicability conditions were originally developed for convex polyhedra. In the nonconvex case they constitute necessary but not sufficient conditions for contact. Thus, they give rise to a conservative strategy, i.e., some pairings which could never realize the minimum distance may not be eliminated. However, in practice this preprocessing permits reducing the computational effort significantly.

3.2 Correctness of the lower distance bound after orientation-based pruning

The key idea is to consider the different types of closest features between the polyhedra. For each case, we prove that there exists an applicable contact and that the corresponding edge-face pairing realizes the minimum distance (i.e., it contains the closest features between the polyhedra). Obviously, function (1) applied to such edge-face pairs provides a lower bound on the actual distance.

The four cases of closest features depicted in Figure 6 are considered:

are described in detail elsewhere.[26,28]

- **Vertex-face**. Clearly, v and f are applicable, and any edge-face pair arising from this applicable contact realizes the minimum distance.

- **Edge-edge**. The same applies to this case: the closest edges are necessarily applicable, and any choice of edge-face pair will realize the minimum distance.

- **Vertex-edge**. Consider the case where v is neither applicable to f_1 nor to f_2. It has to be proven that edge e is applicable to at least one of the adjacent edges of v. This can be shown by projecting these features onto a plane perpendicular to e. The line that joins v and the closest point on e is parallel to such a plane (see Figure 7). Any of the edges adjacent to v that are on the boundary of the projection (e_i or e_j) can be brought into contact with e by performing a translation, i.e., they are applicable to e. Conversely, if e is not applicable to any edge adjacent to v, then v is necessarily applicable to f_1 or f_2, as can be seen by projecting on the same plane. In both cases, any edge-face pair derived from these applicable contacts contains the closest features v and e.

- **Vertex-vertex**. Figure 8 depicts a situation where neither v_1 is applicable to any adjacent face of v_2, nor vice versa. If these polyhedra are brought into contact by a translation, in the neighborhood of v_1 and v_2 the contact will clearly be of edge - edge type, which means that these edges are applicable. Whether this is the case, or one of the vertices is applicable to an adjacent face of the other vertex, the derived edge-face pairs clearly will always contain v_1 and v_2.

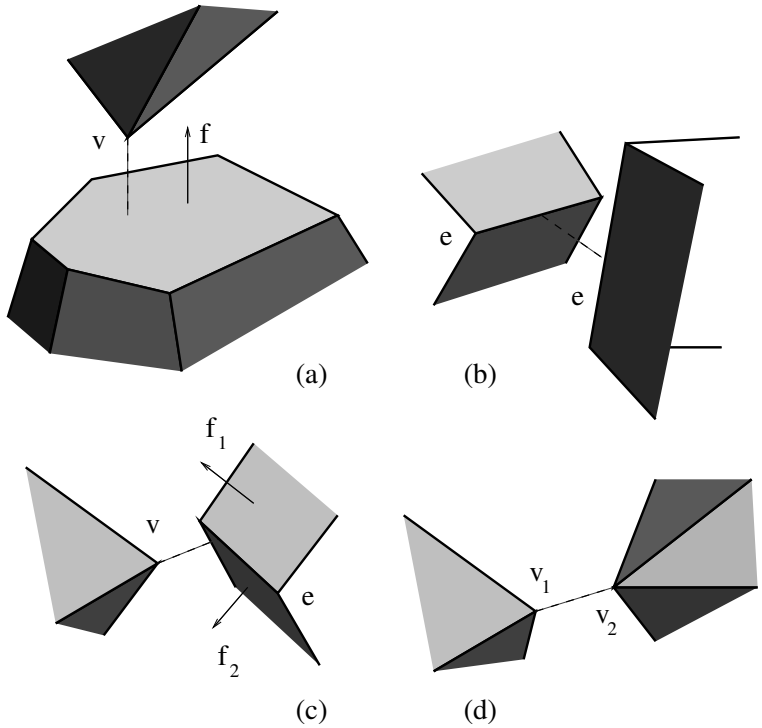

Figure 6: *Closest features:* (a) *vertex-face,* (b) *edge-edge,* (c) *vertex-edge,* (d) *vertex-vertex.*

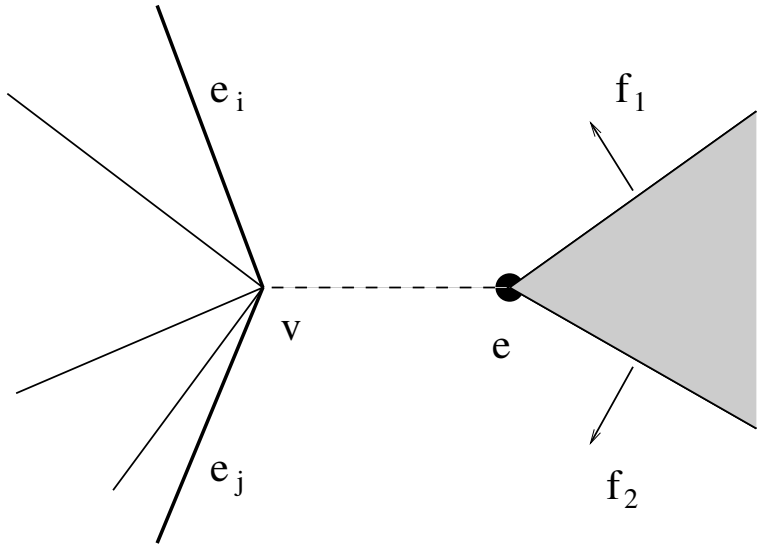

Figure 7: *Closest vertex - edge pair. The vertex v is not applicable to f_1 nor to f_2, but the contacts of e_i and e_j with e are clearly applicable.*

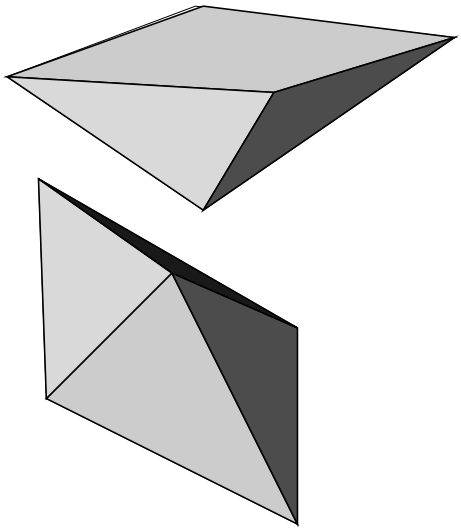

Figure 8: *Closest vertex - vertex pair. If no face is applicable to v_1 nor to v_2, then applicable contacts must exist between the adjacent edges.*

3.3 Experimental results: Enhancement of the lower distance bound

Applicability pruning relies on the geometry and relative orientation of the polyhedra, whereas the lower bound depends also on their relative position. As positional information is not taken into account in the pruning step, nothing can be said in general about the quality of the resulting lower bound. Obviously, pruning might eliminate those edge-face pairings that lead to very bad lower bounds, so that in any case the resulting lower bound will not be worse than before pruning, but the extent of the improvement cannot be quantified theoretically. Therefore, experiments have been undertaken in order to evaluate this improvement.

Experiments carried out on a set of different polyhedra (ranging from the simple tetrahedron to a polyhedral approximation of the sphere with 256 triangular faces) show both an improvement on the computational effort needed to compute the lower distance bound,

as well as the attainment of a better quality in the distance bound (see Figures 9, 10 and 11). The polyhedra have been located at different positions, with diverse relative orientations, thus attaining slightly different performances. However, the results concerning execution time (Figure 10) have been averaged.

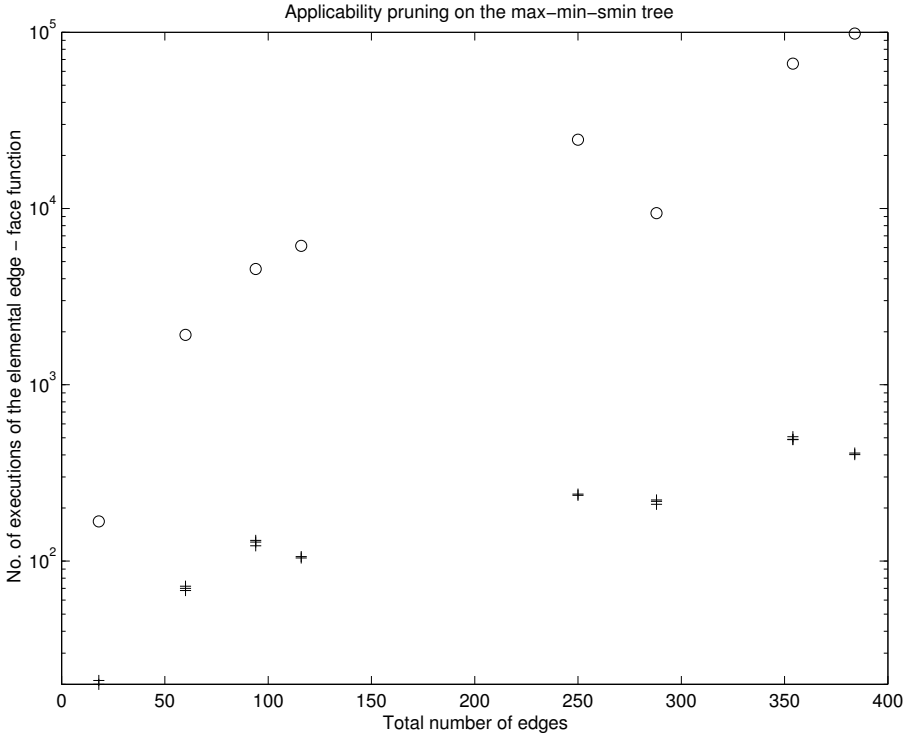

Figure 9: *The number of executions of the elementary edge-face ldb-function for all edge-face pairings (o) and for those pairings resulting from applicable contacts (+). Note that in this, and in the two subsequent figures, a logarithmic scale is used.*

4 Future work: Integration with volumetric and spatial strategies

We have tested the potential of our orientation strategy by comparing its performance with that of a standard distance computation package like PQP. Note that the comparison

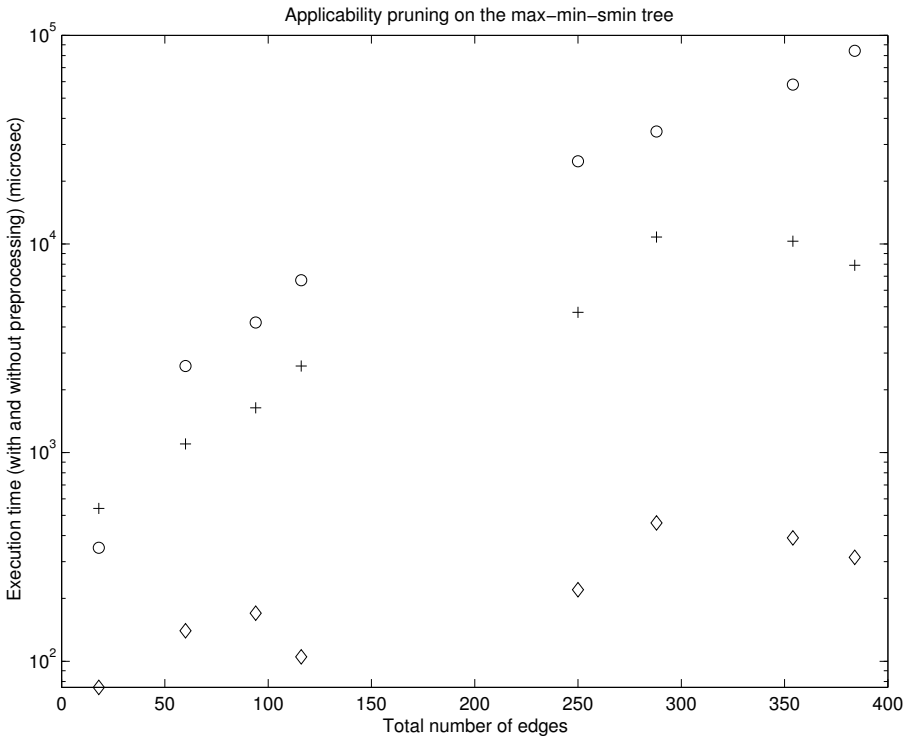

Figure 10: *Comparison of execution times of the algorithms that determine the lower distance bound when only the α-β pruning strategy is applied (o) and when also an applicability pruning preprocessing is performed (both total time ($+$) and time excluding preprocessing ($\diamond$) are displayed). Note that, even including the time needed for preprocessing, the perfomance is much better if applicability pruning is performed. Experiments were carried out on a Sun Ultra 80 workstation.*

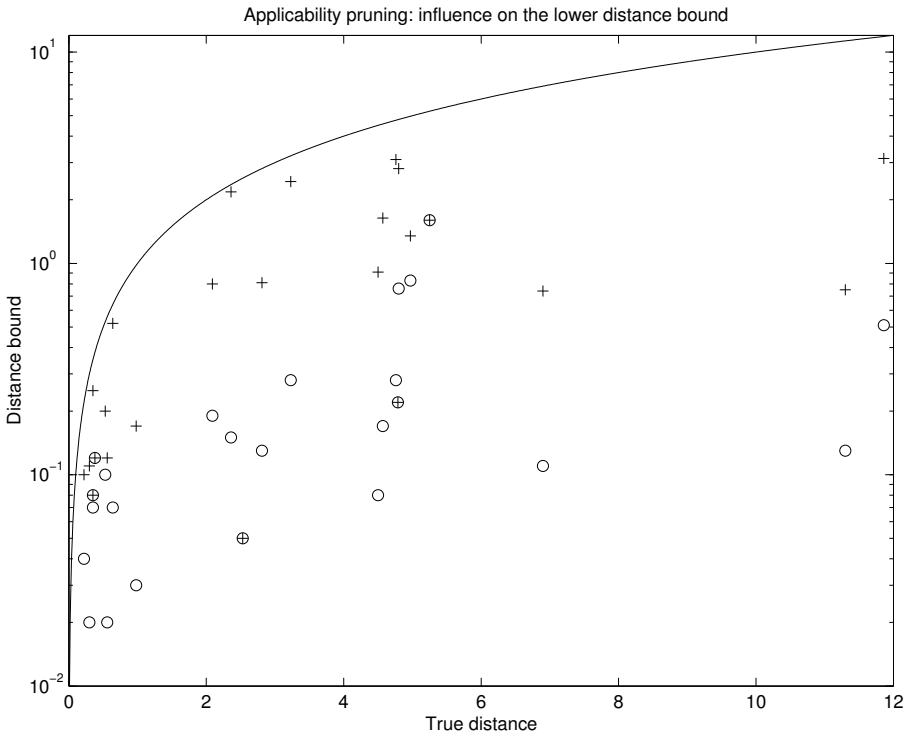

Figure 11: *Comparison of the lower distance bound vs. the real distance between the solids. Although the quality of the distance bound depends mainly on the geometry of the objects, it is also clear that, in general, the closer the objects are, the better the quality of the distance bound obtained. In some cases, the lower distance bounds obtained with (+) and without (o) applicability pruning coincide, but the general tendency is a much better performance in the former case. The mean improvement on distance estimation is of 22.4% in the considered distance range. The continuous line has been drawn to show where the bound would coincide with the true distance.*

is very unbalanced in the sense that our algorithm doesn't benefit from the bounding volume hierarchy (this is the goal of future work, as described in subsection 4.2), while PQP takes full advantage of it. The aim of these preliminary experiments is to assess the promise of our proposed strategy: if its efficiency falls orders of magnitude below PQP, then there is no point in combining it with bounding volume hierarchies, but if their costs are comparable, then it makes sense to try to combine our algorithm with such hierarchies in order to yield a competitive lower distance bound algorithm. In the second subsection, we provide some hints on how we plan to address such integration.

4.1 Promise assessment: experiments in isolation

The potential of the proposed strategy becomes evident when we compare the performance of our algorithm with a state-of-the-art publicly available distance computation package, PQP v1.2,[12] in a setting that involves many nearly touching features. In particular, we have tested the distance computation part of both algorithms (i.e., without considering neither the preprocess of applicability pruning performed in our algorithm, nor the triangulation and the bounding volume hierarchy construction needed in PQP) in the case where polyhedron A shown in Figure 12 is inserted in polyhedron B, without touching it.

Table 1 shows the runtimes of both algorithms for different complexities (the number of protuberances of A and holes in B), where A and B are almost aligned and at different relative positions. It also displays the lower distance bound obtained at each position and the real distance computed by PQP. Despite the unfairness of the comparison, as mentioned above, the runtimes are slightly favourable to our algorithm, although of course

only a lower bound instead of the true distance is obtained.

Note the high stability in execution time of our algorithm for each polyhedra complexity, whereas that for PQP shows larger variations, due to its dependence on the portion of the hierarchy that needs to be traversed. These experiments do also illustrate that *ldb* is a better bound the closer the objects are, as already mentioned in discussing Figure 11. An explanation is that, in certain positions, some very low basic type-A or type-B functions may determine the value of the lower bound, and the distortion becomes more relevant if the objects are distant. This also enforces the interest of combining our approach with a bounding volume hierarchical strategy.

PQP offers the possibility of computing whether the minimum distance falls above a given tolerance or not. If it does, the computation can generally stop without having to traverse the whole hierarchy up to the leaves: if the minimum distance computed between two nodes at a given level exceeds the tolerance, the corresponding subhierarchies that lie below those nodes can be discarded, as the minimum distance between two bounding volumes is always greater or equal than the minimum distance between the enclosed primitives. However, as shown by the experimental results in Table 2, this pruning effect is not attained in these settings where objects are in close proximity: distances between the bounding volumes along the hierarchy are lower than the tolerance, in general, as there are many features that are fairly close between the two polyhedra. By looking at the central column in Table 1, one can observe that, in these settings, a major pruning is obtained when the tolerance is larger than the minimum distance, as very soon two leaves (two triangles) can be found that are closer than the tolerance and this result is exploited to cut off whole subtrees in the hierarchy search recursion process. But in this case the

tolerance acts as an upper bound: it can be ensured (possibly very fast) that the distance is below the tolerance, but not to which extent (the objects may even be colliding, in the extreme case). On the contrary, our distance bound is always a lower bound (although sometimes not a very tight one), naturally leading to a conservative strategy if used in a collision detection context.

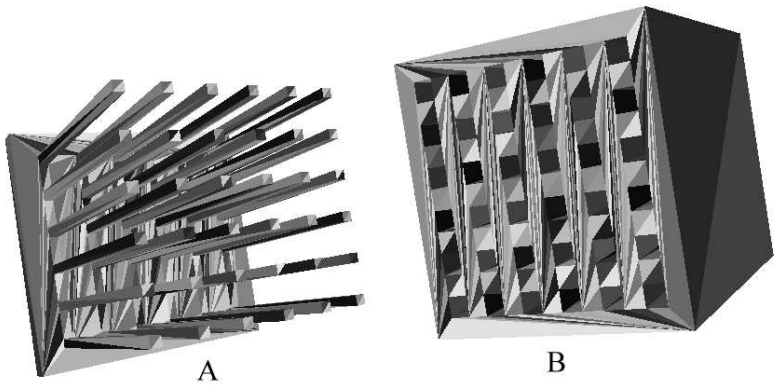

Figure 12: *Polyhedron A with 6x6 protuberances is tested against polyhedron B (6x6 holes) in an insertion position. The polyhedra are displayed after the triangulation*[29] *that has to be performed as a preprocess in order for the polyhedra to be a suitable input for PQP, which is not necessary in our algorithm. Observe that this generic triangulation algorithm creates very narrow triangles in this particular setting.*

These results show that the performance of LDB is comparable to standard volumetric hierarchy based packages, even if the newest ones, like SWIFT++,[15] may increase efficiency by one order of magnitude. Thus, LDB is a good candidate for integration with volume bounding approaches.

4.2 Some hints on a possible integration

Bounding volume hierarchies may incorporate information about the face normals in order to speed up proximity queries. In,[13] the geometry at each node in a standard bounding

complexities	LDB	PQP (Distance)	PQP (Tol=1.01d)	PQP (Collision)	PQP (Tol=0.99d)
3x3	1.8 (0.0250)	14.4 (0.0388)	0.9	9.2	13.7
4x4	5.0 (0.0320)	30.1 (0.0378)	3.1	19.7	29.3
5x5	10.3 (0.0320)	58.1 (0.0371)	7.5	39.3	56.8
6x6	20.3 (0.0118)	95.0 (0.0367)	3.1	63.9	93.6
3x3	1.8 (0.0298)	14.3 (0.0433)	0.8	9.2	13.9
4x4	4.8 (0.0310)	30.1 (0.0423)	3.1	19.4	28.8
5x5	10.4 (0.0309)	60.2 (0.0416)	2.0	38.6	59.4
6x6	20.2 (0.0231)	92.4 (0.0412)	3.0	63.0	90.8
3x3	1.8 (0.0724)	14.8 (0.0859)	0.9	9.1	14.3
4x4	4.9 (0.0209)	30.5 (0.0849)	1.3	19.8	29.5
5x5	10.5 (0.0049)	59.5 (0.0842)	2.0	38.3	56.9
6x6	19.9 (0.0005)	95.8 (0.0838)	3.1	62.3	93.5
3x3	1.8 (0.0073)	16.0 (0.1569)	1.0	9.0	15.4
4x4	4.9 (0.0039)	33.5 (0.1559)	1.4	19.1	31.9
5x5	10.3 (0.0071)	65.5 (0.1552)	2.1	37.7	62.9
6x6	19.5 (0.0073)	101.4 (0.1548)	3.1	61.7	99.5

Table 1: *Runtimes (in milliseconds) and distances (between parentheses) of our lower distance bound computation algorithm (LDB) and PQP 1.2 exact distance computation algorithm, plus execution times of two tolerance verifications, where the first one corresponds to a tolerance of* 1% *over the real distance (thus also collision detection times are shown, see the text) and the second to* 1% *below. The algorithms have been executed on a Sun Ultra 80 workstation (2 ULTRASPARC II processors at 450 MHz, 1Gb RAM), for different complexities of the polyhedra, at four different positions. Only the distance computation part is taken into account, i.e., neither the orientation-based preprocessing, nor the triangulation and building up of the hierarchical representations are included. A standard generic triangulation*[29] *has been employed for the polyhedra in PQP.*

complexities	PQP (Tol=0.90d)	PQP (Tol=0.50d)	PQP (Tol=0.10d)	PQP (Tol=0.01d)	PQP (Tol=0.001d)
5x5	56.6	55.7	55.2	55.0	54.9
6x6	93.3	91.3	86.7	85.5	85.0
5x5	59.0	58.0	57.3	57.0	57.0
6x6	90.8	88.6	86.1	85.5	85.5
5x5	58.1	54.6	53.3	53.3	52.8
6x6	93.5	88.1	85.0	84.6	84.4
5x5	62.7	57.9	53.4	53.4	53.0
6x6	98.5	90.3	84.5	84.2	83.9

Table 2: *Runtimes (in milliseconds) of PQP 1.2 tolerance verification algorithm, for different tolerances. Observe that in this particular setting execution times are very similar, due to very small distances between bounding volumes down to the leaves of the hierarchy.*

volume hierarchy (that in the PQP library) is bounded by a sphere, and a spatialized normal cone is computed for each node (the axis is set to the average normal, and the cone angle is equal to the maximum angle between the axis and the normals of the faces contained in the node). The spheres are used at runtime to compute the *dual view cone* between two given nodes, and the corresponding normal cones are tested for overlap with the dual view cone: if no overlap exists, the corresponding portions of the solids cannot realize a local minimum distance. This information allows to prune away entire subtrees from further consideration, which may not have been possible using only spatial information.

This strategy inspires a possible integration scheme of applicability constraints into a bounding volume hierarchy, by associating feature normals to each node in a straightforward way (i.e., a face of a polyhedron is represented by its normal, or a point on the

unit sphere of orientations, an edge by an arc, and a vertex by a spherical region, again see[26, 28] for details). However, some important differences exist between normal cones and applicability constraints, which have to be taken into account. The first one concerns complexity: whereas the spatialized normal cones approach involves all the time the same number of entities (two cones and two spheres), the applicability constraints approach implies that the deeper the nodes lie in the hierarchy, the less feature normals have to be tested for intersection (dot-region and arc-arc intersections, involving the feature normals of the two considered nodes in the hierarchies). Another difference is that the applicability constraints remain constant as long as the relative orientation of the polyhedra does not change (and even for slight changes in the orientation, more on this later), whereas the dual view cones have to be recomputed also if the relative position changes. This suggests two possible strategies: to compute all feature normal intersections at the root nodes and then propagate the results (i.e., discarding pairings between nodes of the hierarchies that are not linked to common intersections), or to compute these intersections only at the deeper levels of the hierarchies, where only few feature normals are involved. The first option entails the burden of updating the information concerning applicability of contacts along the entire hierarchy when significant orientation changes occur, whereas the second implies to determine at which depth the tradeoff between the cost of determining applicable constraints and the possible pruning of nodes will maximize the savings.

Another possibility consists in integrating the orientation-based pruning in a spatial partition scheme, like octrees or BSP-trees.[30–32] Further subdivisions of given regions, and the corresponding proximity queries, can be avoided if the portions of the objects that are inside these regions display no applicable contacts. Again, to determine at which level of

space subdivision (probably dependent on the involved sizes of the remaining portions of the polyhedra) it is worthwile to consider applicability constraints is not a trivial issue and deserves further research. The widely used concepts of spatial and temporal coherence should also be considered when designing a smart procedure for updating the applicable constraints.

5 Conclusions

A well-established interference detection predicate has been turned into a lower distance bound by just replacing basic predicates and boolean connectors by corresponding continuous functions. The correctness of this bound has been proven both with and without a preprocessing step to lower the cost of computing it.

The preprocessing exploits the relative orientation of the polyhedra to discard features which cannot realize the minimum distance. Thus, it is complementary to the usual approaches based on bounding volume hierarchies.

Experimental results have shown that the mentioned orientation-based preprocessing not only speeds up the computation, but also improves the result, i.e. a tighter lower bound is obtained.

The results of the preprocessing are valid for a given relative orientation of the polyhedra. If for some reason the lower distance bound has to be recomputed after a period of time where the orientations may have changed, the preprocessing has to be performed again. This limitation may be overcome by further research. If the movements' law is known in advance, this can be done by determining a good parameterization of the ro-

tational part. Then, an efficient way to subdivide the domain of this parameter has to be found, such that the set of applicable contacts remains unchanged in each one of the intervals resulting from the subdivision.

The possibility of combining an orientation-based strategy like the one presented here with a hierarchy of bounding volumes, like in PQP or SWIFT++, or a spatial partition scheme, exploiting their benefits both in distant as well as in nearly touching situations, deserves further research.

Acknowledgments

This research has been partially supported by the Spanish Science and Technology Commission (CICYT) under contract TAP99-1086-C03-01 (project "Constraint-based computation in robotics and resource allocation") and the Catalan Research Commission, through the "Robotics and Control" group.

References

1. M. C. Lin and S. Gottschalk, Collision detection between geometric models: a survey, in IMA Conference on Mathematics of Surfaces, 1, San Diego (CA), 1998 pp. 602–608.

2. P. Jiménez, F. Thomas, and C. Torras, Collision detection: a survey, Computers and Graphics, 25 (2001) (2), 269–285.

3. S. A. Cameron, T. Fenton-May, and J. Pitt-Francis, Directions in the use and computation of proximity queries, in Proc. IEEE Geometric Modeling and Processing

-Theory and Applications, Wako, Saitama (Japan), 2002 pp. 43–52.

4. J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi, I-collide: An interactive and exact collision detection system for large-scale environments, in Proceedings of ACM Int. 3D Graphics Conference, 1, 1995 pp. 189–196, http://www.cs.unc.edu /~geom /I_COLLIDE.html.

5. E. G. Gilbert, D. W. Johnson, and S. Keerthi, A fast procedure for computing the distance between complex objects in three dimensional space, IEEE J Robotics Automat, 4 (1988) (2), 193–203.

6. E. G. Gilbert and C.-P. Foo, Computing the distance between general convex objects in three-dimensional space, IEEE Trans Robotics Automat, 6 (1990) (1), 53–61.

7. E. G. Gilbert and C. J. Ong, New distances for the separation and penetration of objects, in Proc. IEEE Int. Conf. on Robotics and Automation, 1, San Diego (CA), 1994 pp. 579–586.

8. M. C. Lin and J. F. Canny, A fast algorithm for incremental distance calculation, in Proc. IEEE Int. Conf. on Robotics and Automation, 2, Sacramento (CA), 1991 pp. 1008–1014.

9. S. A. Cameron, Enhancing gjk: Computing minimum and penetration distances between convex polyhedra, in Proc. IEEE Int. Conf. on Robotics and Automation, Albuquerque (NM), 1997 pp. 3112–3117.

10. B. Mirtich, V-clip: Fast and robust polyhedral collision detection, ACM Transactions on Graphics, 17 (1998) (3), 177–208, http://www.merl.com/projects/vclip/.

11. S. A. Ehmann and M. C. Lin, Accelerated proximity queries between convex polyhedra by multi-level voronoi marching, in IEEE IROS, 1, Takamatsu (Japan), 2000 pp. 2101–2106.

12. E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, Fast proximity queries with swept sphere volumes, Technical Report TR99-018, Dep. of Comp. Sci., UNC Chapel Hill, 1999, http://www.cs.unc.edu/~geom/SSV/.

13. D. E. Johnson and E. Cohen, Spatialized normal cone hierarchies, in ACM Symposium on Interactive 3D Graphics, ACM SIGGRAPH, 2001 pp. 129–134, URL `citeseer.nj.nec.com/446423.html`.

14. K. Kawachi and H. Suzuki, Distance computation between non-convex polyhedra at short range based on discrete voronoi regions, in Proc. IEEE Geometric Modeling and Processing -Theory and Applications, Hong Kong (China), 2000 pp. 123–128.

15. S. A. Ehmann and M. C. Lin, Accurate and fast proximity queries between polyhedra using convex surface decomposition, in EUROGRAPHICS, 20, Manchester (UK), 2001 pp. 500–510.

16. E. Rimon and S. P. Boyd, Efficient distance computation using best ellipsoid fit, in Proc. 1992 Int. Symp. on Intelligent Control, Glasgow, Scotland (UK), 1992 pp. 360–365.

17. S. A. Cameron, Approximation hierarchies and s-bounds, in ACM Symp. on Solid Modelling and Applications, 1, Austin (TX), 1991 pp. 129–137.

18. S. A. Cameron, A comparison of two fast algorithms for computing the distance between convex polyhedra, IEEE Trans Robotics Automat, 13 (1997), 915–920.

19. S. Quinlan, Efficient distance computation between non-convex objects, in Proc. IEEE Int. Conf. on Robotics and Automation, 4, San Diego (CA), 1994 pp. 3324–3329.

20. K. Zikan and P. Konečný, Lower Bound of Distance in 3D, Technical Report FIMU-RS-97-01, Faculty of Informatics, Masaryk University, Brno, 1997.

21. D. E. Johnson and E. Cohen, A framework for efficient minimum distance computations, in Proc. IEEE Int. Conf. on Robotics and Automation, Leuven, Belgium, 1998 pp. 3678–3683, URL `citeseer.nj.nec.com/johnson98framework.html`.

22. E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, Fast distance queries with rectangular swept sphere volumes, in Proc. IEEE Int. Conf. on Robotics and Automation, 4, San Francisco (CA), 2000 pp. 3719–3726.

23. J. F. Canny, Collision detection for moving polyhedra, IEEE Trans Patt Anal Mach Intell, 8 (1986) (2), 200–209.

24. B. R. Donald, A search algorithm for motion planning with six degrees of freedom, Artificial Intelligence, 31 (1987) (3), 295–353.

25. F. Thomas and C. Torras, Interference detection between non-convex polyhedra revisited with a practical aim, in Proc. IEEE Int. Conf. on Robotics and Automation, 1, San Diego (CA), 1994 pp. 587–594.

26. P. Jiménez and C. Torras, An orientation-based pruning tool to speed up interference detection between translating polyhedral models, International Journal of Robotics Research, 20 (2001) (6), 466–483.

27. P. Jiménez, Static and dynamic interference detection between nonconvex polyhedra, Ph.D. thesis, Universitat Politècnica de Catalunya, 1998, http://www-iri.upc.es/people/jimenez/phdthesis.html.

28. P. Jiménez and C. Torras, Speeding up interference detection between polyhedra, in Proc. IEEE Int. Conf. on Robotics and Automation, 2, Minneapolis (MN), 1996 pp. 1485–1492.

29. A. Narkhede and D. Manocha, Fast polygon triangulation based on Seidel's algorithm, in Graphics Gems 5, edited by A. Paeth, Academic Press, 1995 pp. 394–397, http://www.cs.unc.edu/~dm/CODE/GEM/chapter.html.

30. M. Held, J. T. Klosowski, and J. Mitchell, Evaluation of collision detection methods for virtual reality fly-throughs, in Proc. Seventh Canadian Conf. Computer Geometry, 3, 1995 pp. 205–210.

31. B. F. Naylor, J. A. Amatodes, and W. C. Thibault, Merging bsp trees yields polyhedral set operations, in Computer Graphics (SIGGRAPH90 Proc.), 24, Dallas (TX), 1990 pp. 115–124.

32. S. Ar, B. Chazelle, and A. Tal, Self-customized BSP trees for collision detection, Computational Geometry, 15 (2000) (1-3), 91–102, URL `citeseer.nj.nec.com/ar00selfcustomized.html`.