Interference Detection between Non-Convex Polyhedra Revisited with a Practical Aim *

Federico Thomas and Carme Torras

Institut de Cibernètica (UPC – CSIC)
Diagonal 647, 2 planta
08028 Barcelona, SPAIN

e-mail: thomas@ic.upc.es torras@ic.upc.es

Abstract

well-known that exact *interference checking between arbitrary polyhedra has $O(mn)$ complexity, where m n are the number of edges in the two polyhedra. This st a worst-case bound that still leaves plenty of room lgorithm improvement in practice.*

he algorithm presented herein has been developed so :

Minimize the number of operations *that each pair- of edges entails (i.e. the constant factor k in the t-case complexity $O(nm) = knm$). We prove that factor is 4.5 for multiplications and 8.5 for addi- s.*

Avoid the construction of auxiliary geometric enti- *The standard approach is to decompose the non- ex polyhedra (or their faces) into convex entities then check for interference in this convex setting. entails the construction of many ficticious edges faces, which indirectly contribute to the growth of factor above. Our algorithm reduces interference ction to the additive combination of determinants lving only vertices of the polyhedra.*

Permit the straightforward application of prunning egies *to most practical situations, so that the worst- bound above is reached only when truly needed.*

Allow the derivation of both directional and undi- onal distance bounds *between the polyhedra, which e extremely useful for collision avoidance and local planning.*

he simplicity and homogeneity of the algorithm has o a quick implementation, which has been proven to bust and fast. Some performance measurements are rted.

his research has been partially supported by the ESPRIT asic Research Actions Program of the EC under contract 546 (project PROMotion).

1 Introduction

Interference detection algorithms lie at the base of many applications in Robotics, CAD-CAM and Computer Graphics. To name but a few, local path planning, assembly verification and computer animation are examples of tasks relying on such algorithms. Since interference detection needs to be performed in a repetitive way in those applications, every possible effort should be devoted to speeding this basic operation up.

One way to attain this speed up is to enclose each object within either a simpler approximating solid (typically, "an enclosing box"), or a hierarchy of solids. Since spheres are rotationally invariant and thus their interference can be easily checked [13], they have been extensively used. Two approaches based on overlapping [17, 20] and dynamic [25] spherical descriptions have been proposed.

Approximate and hierarchical representations are particularly helpful in dealing with non-cluttered environments. However, in tasks with small tolerances (such as, for instance, assembly planning and validation [24]) eventually it becomes necessary to work with the most exact representation of objects available.

The two most widely used representation schemes are Constructive Solid Geometry (CSG) and Boundary representations (B-rep). In the context of CSG representations, exact interference detection is performed through the so-called "null object detection" algorithms [19]. Ways to speed up the application of these algorithms have been devised as, for instance, setting a set of convex bounds around the primitive nodes and, through simple composition rules, isolating zones of space where objects might interfere [6].

For several reasons, however, most solid modelling and robot simulation environments resort to B-rep's when performing exact interference detection. When the objects can be modelled as convex polyhedra, effi-

s algorithms developed in the field of Computational metry can be readily applied [9, 15]. In the case non-convex polyhedra need to be used, the stan- approach consists in decomposing the polyhedra ts faces) into convex entities [7, 12]. Each entity one polyhedron is then tested for intersection with entity from the other, until either an intersection rs or all entities from both polyhedra are free from section.

his requires a preprocessing of the polyhedra in order e decomposed, leading to the creation of many "ficti- s" edges and faces, which are subsequently tested for ference. As a result, although the worst-case com- ty remains $O(nm)$, since the number of ficticious s is linear in the number of real edges, the constant plexity factor k associated to each pairing of real s grows steadily with the "degree" of non-convexity e polyhedra.

the algorithm presented herein, *k is truly constant*, e sense that it does not depend on the "degree" of convexity of the polyhedra. Moreover, our aim has to keep this constant as small as possible and, ac- y, we prove that it is 4.5 for multiplications and 8.5 dditions. Note that this is a constant upper bound ch is nice to have, instead of having one dependent e complexity of the input), and thus we have imple- ted some prunning strategies in order to make the al complexity decrease when the simplicity of the tion permits.

e algorithm falls in the tradition of predicate-based oaches [2, 3, 7] and, therefore, it is easy to derive it a way to compute distance bounds between the hedra. Furthermore, the elements from the polyhe- ealizing the minimum distance and those near this mum can be directly provided by the algorithm. facility is very handy for collision-avoidance and its real-time animation, since the critical zones in cene where collisions may occur can be continuously itored.

s a final practical remark, let us mention that some- s intersection construction algorithms are used to ct interferences, mainly because solid modellers usu- provide this facility. It goes without saying that struction" algorithms are always more expensive their "detection" counterparts.

e paper is structured as follows. Section 2 intro- s some notation used throughout this paper. Sec- 3 gives a brief description of the functions and pred- s associated with the basic contacts between two hedra. In Section 4 the algorithm for interference tion is fully described. The detailed complexity of lgorithm is worked out in Section 5. Section 6 tack- he problem of obtaining distance bounds between hedra. Implementation details, prunning strategies and experimental results are reported in Section 7, and Section 8 summarizes the main points in this paper and discusses further research.

2 A few definitions

Definition 1. *Tangent vector in* $\Re^3$. A tangent vector in $\Re^3$ is a pair $(v, p) \in \Re^3 \times \Re^3$, where v is a vector applied at point p.

According to this definition, points, lines and planes can be represented as follows. A point i can be represented through a vector v_i from the origin to the point; a line by means of a tangent vector (e_i, p_i), e_i being a unit vector along the line and p_i a point on it; and a plane i through a tangent vector (f_i, p_i), f_i being a vector normal to the plane and p_i any point on the plane. Nevertheless, once the adjacency relationships between the elements of a polyhedron – vertices, edges and faces – are known, a single vector is enough to represent each of them. Thus, it is possible to define the dot product $(\langle , \rangle)$, the cross product $(\times)$ and the determinant $(|, , |)$, between vertices, edges and faces.

Definition 2. *Boundary and coboundary operators.* ∂ will denote the boundary operator. If f_k is a vector normal to a face k, ∂f_k will denote the ring of edges around the face clockwise ordered from outside the polyhedron. If e_j is a vector along an edge j, ∂e_j will denote the vertices bounding this edge. The *coboundary* operator is the dual operator and it will be denoted by δ. The coboundary of a vertex is a set of edges converging at this vertex clockwise ordered from outside the polyhedron.

Definition 3. *Halfboundary and halfcoboundary operators.* Given a direction for an edge, an order relationship between the vertices of its boundary and the faces of its coboundary can be established. The adopted convention is as follows:

$$\left.\begin{array}{l} \langle (\delta^- e_j \times \delta^+ e_j), e_j \rangle > 0 \\ \langle (\partial^+ e_j - \partial^- e_j), e_j \rangle > 0 \end{array}\right\} \quad (1)$$

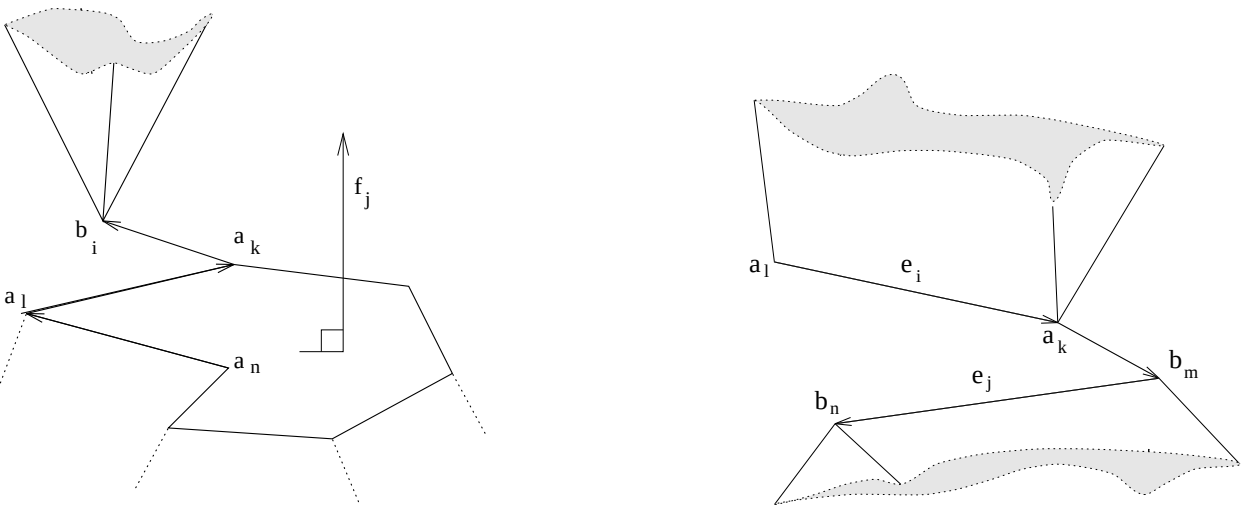
with $\partial e_j = \{\partial^- e_j, \partial^+ e_j\}$ and $\delta e_j = \{\delta^- e_j, \delta^+ e_j\}$, where ∂^- and ∂^+ denote the halfboundary operators and δ^-, and δ^+ the halfcoboundary operators.

3 The two basic contacts

There are two basic contacts between two polyhedra in 3D euclidean space [7, 10]; namely:

Type-A A face of one polyhedron is in contact with a vertex of the other polyhedron.

Type-B An edge of one polyhedron is in contact with an edge of the other polyhedron.



1 *Elements intervening in:* A_{b_i,f_j} **(a)**; *and* B_{e_i,e_j} **(b)**.

is clear that all other contacts can be expressed as mbination of these two basic contacts.
ccording to *fig. 1a*, we define

$$A_{b_i,f_j} =< f_j, b_i - a_k >, \quad \text{for any } a_k \in \partial^2 f_j. \qquad (2)$$

will be shown that the sign of (2) is irrelevant for our oses (we will only look for sign changes). Thus, the nal to the face can be pointing outward or inward olyhedron it belongs to.
vertex b_i meets the plane supporting face f_j, then $_j = 0$. We also define the predicate $\mathbf{A}_{b_i,f_j}$, as- ted with the function A_{b_i,f_j}, which is true when $_j > 0$, and false otherwise.
kewise, according to *fig. 1b*, we define

$$_{e_j} =< e_i \times e_j, b_m - a_k >, \quad a_k = \partial^+ e_i \ , b_m = \partial^- e_j. \qquad (3)$$

hus, if the line supporting edge e_i meets the line orting edge e_j, then $B_{e_i,e_j} = 0$. We also define redicate $\mathbf{B}_{e_i,e_j}$, associated with the function B_{e_i,e_j}, h is true when $B_{e_i,e_j} > 0$, and false otherwise. As e, only sign changes of these functions are relevant ur purposes.

Algorithm for interference detection

ace intersection of two plane-faced objects occurs if only if an edge of one object intersects a face of the r.
r an edge e_0 to intersect a face f_1 it is necessary its two endpoints lie on different sides of the plane upporting face f_1, i.e.

$$(\mathbf{A}_{\partial+e_0,f_1} \oplus \mathbf{A}_{\partial-e_0,f_1}) \qquad (4)$$

is true, $\oplus$ being the exclusive or operator (*XOR*, for short) defined as $(a \oplus b) = (a \cap \overline{b}) \cup (\overline{a} \cap b)$.

Assuming that the above predicate holds, let us refer to *fig. 2* for further discussion. Let Π_0 be a plane containing edge e_0 which, for convenience, we will assume to be the plane supporting a face f_0, such that $f_0 \in \delta e_0$. The line supporting e_0 divides Π_0 into two half planes, Π_0^- and Π_0^+. Now the number of edges of the face piercing each of these half planes (which determines whether the edge intersects the face) can be obtained as follows. Let $\mathcal{E}^+$ be the set of edges piercing plane Π_0 and pointing *upwards* and $\mathcal{E}^-$, those piercing the same plane and pointing *downwards*. So that

$$\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^- = \left\{ e_j \in \partial f_1 \mid \mathbf{A}_{\partial+e_j,f_0} \oplus \mathbf{A}_{\partial-e_j,f_0} \right\}. \qquad (5)$$

Then, the function B_{e_0,e_j}, $e_j \in \mathcal{E}^+$, is positive iff the intersection of e_j and Π_0 is located on Π_0^-, and negative iff it is located on Π_0^+. Likewise, the function $B_{e_0,-e_j}$, $e_j \in \mathcal{E}^-$, is positive iff the intersection of e_j and Π_0 is located on Π_0^-, and negative iff it is located on Π_0^+. This can be easily seen by considering that $\mid a, b, c \mid$ is positive iff vectors a, b, and c define a right-handed coordinate system.

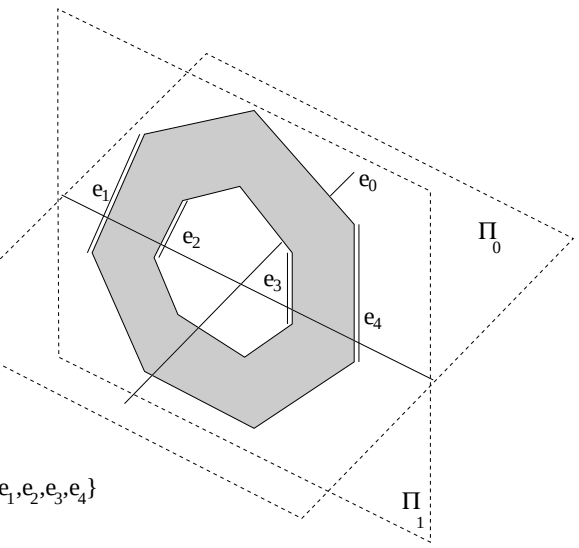
Thus, the number of edges piercing half plane Π_0^+ (or half plane Π_0^-) is odd iff

$$\bigoplus_{e_k \in \mathcal{E}} \left[\left(\mathbf{A}_{\partial+e_k,f_0} \cap \mathbf{B}_{e_0,e_k} \right) \cup \left(\mathbf{A}_{\partial-e_k,f_0} \cap \mathbf{B}_{e_0,-e_k} \right) \right] =$$

$$\bigoplus_{e_k \in \mathcal{E}} \left(\mathbf{A}_{\partial-e_k,f_0} \oplus \mathbf{B}_{e_0,e_k} \right) \qquad (6)$$

is true, or in other words,

$$\bigoplus_{e_j \in \partial f_1} \left(\mathbf{A}_{\partial+e_j,f_0} \oplus \mathbf{A}_{\partial-e_j,f_0} \right) \cap \left(\mathbf{A}_{\partial-e_j,f_0} \oplus \mathbf{B}_{e_0,e_j} \right) \qquad (7)$$



2 *Detecting the intersection between face f_1 and edge Here the first part of predicate (8) is true, but the second is not.*

ue. Thus, the conjunction of (4) and (7) leads to a ssary and sufficient condition for edge e_0 to intersect f_1, which can be expressed as:

$$(\mathbf{A}_{\partial+e_0,f_1} \oplus \mathbf{A}_{\partial-e_0,f_1}) \cap$$

$$\bigoplus_{e_j \in \partial f_1} (\mathbf{A}_{\partial+e_j,f_0} \oplus \mathbf{A}_{\partial-e_j,f_0}) \cap (\mathbf{A}_{\partial-e_j,f_0} \oplus \mathbf{B}_{e_0,e_j}) \Big] . \quad (8)$$

otice that basic predicates are always combined ugh *XOR* operators and, since $(a \oplus b) = (\overline{a} \oplus \overline{b})$, *sign changes of the basic predicates are relevant to ct intersections.*

t us now move from surface intersection detection bject intersection detection. Two polyhedra intersif and only if one of the following situations arises (1) an edge of one polyhedron is piercing a face of other polyhedron; or (2) a vertex of one polyhedron side the other polyhedron. A necessary and sufficondition to detect the first situation can easily be ined by iterating the application of predicate (8) for dges of one polyhedron and all faces of the other, vice-versa.

he detection of the second situation can be reduced he problem of checking whether the number of faces ed by an edge determined by the considered vertex a point far enough from the polyhedra is odd or , thus the treatment is exactly the same as for the situation. The predicate that becomes true when ex v_0 is inside the polyhedron P is then:

$$\bigoplus_{f_i \in P} [(\mathbf{A}_{v_0,f_i} \oplus \mathbf{A}_{v_\infty,f_i}) \cap$$

$$\bigoplus_{e_j \in \partial f_i} (\mathbf{A}_{\partial+e_j,f_0} \oplus \mathbf{A}_{\partial-e_j,f_0}) \cap (\mathbf{A}_{\partial-e_j,f_0} \oplus \mathbf{B}_{e_0,e_j})] \quad (9)$$

where v_∞ is a point far from the two polyhedra, e_0 is the edge with endpoints v_0 and v_∞, and f_0 is a ficticious face containing edge e_0.

Therefore, we can obtain a necessary and sufficient condition to detect the second situation by applying the predicate (9) for all vertices of the two polyhedra.

Of course, a careful implementation must take care of degenerate situations, i.e. those in which one of the chosen planes Π_0 contains a vertex of the face against which it is tested. This issue is addressed in Section 6. Another practical issue to consider is the development of prunning strategies to cut down the number of predicates evaluated in most practical situations. Some of these strategies are briefly outlined in Section 7.

5 Detailed complexity

Many basic functions share the same operations and this fact must be taken into account to obtain a tight complexity bound. Common operations can be obtained by redefining basic functions in terms of the locations of vertices of both polyhedra. Actually, type-B functions, B_{e_i,e_j}, can be redefined as (see *fig. 1b*):

$$B_{e_i,e_j} = \begin{vmatrix} a_l & 1 \\ a_k & 1 \\ b_m & 1 \\ b_n & 1 \end{vmatrix} = \begin{vmatrix} a_k \\ b_m \\ b_n \end{vmatrix} - \begin{vmatrix} a_l \\ b_m \\ b_n \end{vmatrix} + \begin{vmatrix} a_l \\ a_k \\ b_n \end{vmatrix} - \begin{vmatrix} a_l \\ a_k \\ b_m \end{vmatrix}, \quad (10)$$

or, in other words,

$$B_{e_i,e_j} = < b_m \times b_n, a_k > - < b_m \times b_n, a_l > +$$

$$< a_l \times a_k, b_n > - < a_l \times a_k, b_m >, \quad (11)$$

where all determinants involve an edge of one polyhedron and a vertex of the other. On the other hand, type-A basic functions, A_{b_i,f_j}, can be redefined as (see *fig. 1a*):

$$A_{b_i,f_j} = \begin{vmatrix} a_n & 1 \\ a_l & 1 \\ a_k & 1 \\ b_i & 1 \end{vmatrix} =$$

$$< a_l \times a_k, b_i > + < a_n \times a_l, b_i > + < a_n \times a_k, a_l - b_i >, \quad (12)$$

which only introduces an extra determinant per function.

All determinants involving an edge of one polyhedron and a vertex of the other require the computation of $(E_P + E_Q)$ cross products, plus $(V_P E_Q + E_P V_Q)$ dot products, where E_P and V_P are respectively the number of edges and vertices of polyhedron P. In addition,

-B functions require $3E_PE_Q$ additions to combine rminants, and type-A functions require the compun of the extra determinants, that is, one cross prodper face, three additions and one dot product for y face-vertex pairing, plus $2(V_PF_Q + F_PV_Q)$ addis to combine determinants, where F_P is the number ces of polyhedron P. Since a cross product entails ultiplications and 3 additions and a dot product en- 3 multiplications and 2 additions, the total number ultiplications and additions required are:

$$6(E_P + E_Q) + 3(V_PE_Q + E_PV_Q) + 6(F_P + F_Q) + 3(F_PV_Q + V_PF_Q), \quad (13)$$

$$3(E_P + E_Q) + 2(V_PE_Q + E_PV_Q) + 3E_PE_Q + 3(F_P + F_Q) + 7(F_PV_Q + V_PF_Q), \quad (14)$$

ectively.

ssuming that both the number of vertices and the ber of faces are approximately half the number of s, the dominating quadratic term for multiplications $5mn$ and that for additions is $8.5mn$, m and n being number of edges in one and the other polyhedron.

Distance bounds and collision detection

simplest approach to detect collisions between movbjects is timeslicing, that is checking for interfer- at fixed time intervals. Unfortunately, collisions be arbitrarily brief and may begin and end between two timeslices. Thus, this approach has an implicit eoff between efficiency (number of timeslices) and dence in the validity of collision detection.

ternative approaches to collision detection through slicing are the swept volume approach and fourensional interference detection [5].

nother approach arises when distances between the cts involved are available. Separation distances are ul both as guiding heuristics for search-based path ning algorithms [11] and for obstacle avoidance alhms based on potential functions [14]. Even if the puted distance between two given objects is smaller agnitude than the actual distance but has the corsign, then a spatial interval where no collision can place is obtained. Using an inexpensive lower nd which requires more timeslices along a path may nore efficient than using an accurate but expensive 26] separation distance which requires less times- . This is the approach taken in [8], in which the nce between enclosing solids is a lower bound on distance between the enclosed solids, and also the oach taken here.

The usual way of computing distances between nonconvex polyhedra relies again on a decomposition into convex pieces [12], and then optimization techniques are used to find the distance between the convex pieces [4]. It would be desirable to compute separation distances, or distance bounds, directly from the representation of general polyhedra used in typical solid modelling systems. In what follows we derive this capability from the predicate for interference detection presented in the preceding section.

When the goal is motion planning, and not just static interference detection, the original real-valued forms that led to the predicates for basic contacts (refer to equations (2) and (3)) become of great interest. They can be easily converted into functions of the relative configuration of the two polyhedra. This configuration consists of a relative position $\mathbf{x} \in \Re^3$ and a relative orientation $\mathbf{q} \in SO(3)$, where $SO(3)$ is the group of three-dimensional rotations. Thus, from (2) and (3), we get two real-valued functions $A_{b_i,f_j}(\mathbf{x},\mathbf{q})$ and $B_{e_i,e_j}(\mathbf{x},\mathbf{q})$.

Note that if f_j in (2) is taken to be a normalized vector, then the function A_{b_i,f_j} provides the euclidean distance between vertex b_i and the plane supporting face f_j. Analogously, if $e_i \times e_j$ in (3) is a normalized vector, then the function B_{e_i,e_j} provides the euclidean distance between the lines supporting edges e_i and e_j.

Now, as was pointed out in [7, § 5.1.1], more complex predicates built upon the two basic ones above can also be transformed into functions by just replacing logical AND by the *min* function and logical OR by the *max* function. The XOR functional form, i.e. that corresponding to $(a \oplus b)$ is therefore:

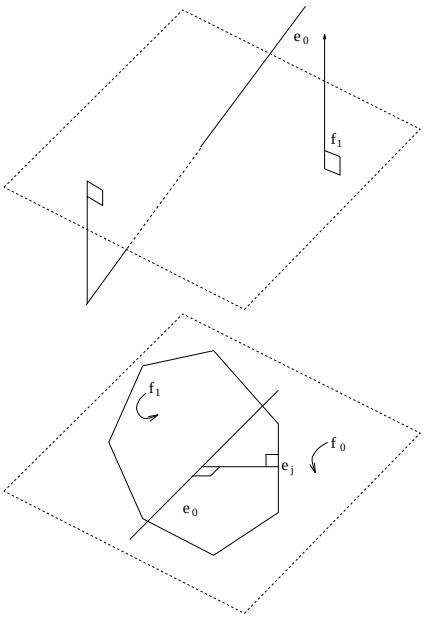
$$max(min(a,-b), min(-a,b)) = min(max(a,b), -min(a,b)) = \begin{cases} +min(|\,a\,|,|\,b\,|), & \text{if } a \cdot b < 0 \\ -min(|\,a\,|,|\,b\,|), & \text{otherwise.} \end{cases} \quad (15)$$

Thus, the functional form corresponding to predicate (4) – which evaluates the **interference between an edge e_0 and the plane supporting a face f_1** – is:

$$min(max(A_{\partial^+e_0,f_1}, A_{\partial^-e_0,f_1}), -min(A_{\partial^+e_0,f_1}, A_{\partial^-e_0,f_1})) = \begin{cases} +min(|\,A_{\partial^+e_0,f_1}\,|,|\,A_{\partial^-e_0,f_1}\,|), & \text{if } e_0 \text{ intersects the plane supporting } f_1 \\ -min(|\,A_{\partial^+e_0,f_1}\,|,|\,A_{\partial^-e_0,f_1}\,|), & \text{otherwise} \end{cases} \quad (16)$$

where the dependency on ($\mathbf{x}$, $\mathbf{q}$) has been dropped to ease the notation.

The geometric interpretation of this function is that it is positive when the edge and the plane intersect and negative otherwise. Moreover, provided f_1 is a normalized vector, the absolute value of function (16) is the euclidean distance from the plane to the closest endpoint



3 (a) *Geometric interpretation of equation (16); and geometric interpretation of (18).*

---

e edge (see *fig. 3a*). In other words, this function actly the minimum translation that the edge would to undergo in order to be just "in contact" with plane without intersecting it, thus providing a very ral measure of the amount of intersection between dge and a plane.

order to transform the predicate (7) – which eval- s the **interference between the line support- edge e_0 and a face f_1** – into a function, we note that the functional form corresponding to $\oplus\, a_2 \oplus \ldots \oplus\, a_n$) is:

$$smin(a_1, a_2, \ldots a_n) =$$

$$\begin{cases} -min(\mid a_1 \mid, \mid a_2 \mid, \ldots, \mid a_n \mid), & \text{if an odd number of } a_i\text{'s are positive} \\ -min(\mid a_1 \mid, \mid a_2 \mid, \ldots, \mid a_n \mid), & \text{otherwise.} \end{cases} \tag{17}$$

hen, the functional form corresponding to predicate s:

$$smin_{e_j \in \partial f_1}(min(\ smin(A_{\partial^+ e_j, f_0}, A_{\partial^- e_j, f_0}),\ smin(A_{\partial^- e_j, f_0}, B_{e_0, e_j}))). \tag{18}$$

he geometric interpretation of this function is that positive when the line and the face intersect and tive otherwise. Moreover, provided f_0 and $e_0 \times e_j$, are normalized vectors, the absolute value of func- (18) is a lower bound on the euclidean distance from ine supporting edge e_0 to the closest edge $e_j \in \partial f_1$ *fig. 3b*).

A lower bound instead of the actual euclidean distance is obtained due to the fact that type-A functions provide vertex-plane distances which are lower bounds on the actual vertex-line distances (from vertices $\partial^+ e_j$, $\partial^- e_j$ to the line supporting edge e_0), as well as the fact that type-B functions provide line-line distances which are lower bounds on the actual line-edge distances (from the line supporting edge e_0 to edges e_j). The minimum of all vertex-line distances and line-edge distances would be the actual euclidean distance between a line and a face.

In sum, the functional form corresponding to predicate (8) is:

$$\begin{aligned} min(\ & smin(A_{\partial^+ e_0, f_1}, A_{\partial^- e_0, f_1}), \\ & smin_{e_j \in \partial f_1}(min(\ smin(A_{\partial^+ e_j, f_0}, A_{\partial^- e_j, f_0}), \\ & \qquad smin(A_{\partial^- e_j, f_0}, B_{e_0, e_j})))). \end{aligned} \tag{19}$$

Analogously, the functional form corresponding to predicate (9) is:

$$\begin{aligned} smin_{f_i \in P}(min(\ & smin(A_{v_0, f_i}, A_{v_\infty, f_i}), \\ & smin_{e_j \in \partial f_i}(min(\ smin(A_{\partial^+ e_j, f_0}, A_{\partial^- e_j, f_0}), \\ & \qquad smin(A_{\partial^- e_j, f_0}, B_{e_0, e_j})))). \end{aligned} \tag{20}$$

Therefore, by taking the *maximum* of function (19) over all edges of one polyhedron and all faces of the other and vice-versa, and then taking the *maximum* again of the value just obtained and that resulting from evaluating function (20) over all vertices of both polyhedra, a lower bound on the amount of intersection between two non-convex polyhedra is obtained.

A lower bound on the distance between two objects is well-behaved if it takes the value zero only when the two objects are in contact, and it grows as fast as possible with the distance. The lower bound we have derived can be tuned to satisfy both conditions.

Note that the lower bound can take the value zero only when the plane supporting face f_0 in (18) contains a vertex of the other polyhedron. The other degenerate situation in which the line supporting edge e_0 meets the line supporting an edge e_j of face f_i, leading to $B_{e_0, e_j} = 0$, does not nullify the whole function because $smin(A_{\partial^+ e_j, f_0}, A_{\partial^- e_j, f_0})$ must be non-zero if we are not in the preceding case. Of course, this situation would never arise if general position is assumed.

Anyway, the quality of the lower bound can be improved (i.e. the lower bound can be made higher and, in particular, non-zero but at a point) by appropriately chosing the plane Π_0 containing edge e_0 (refer to Section 4). In fact, this plane has been taken arbitrarily to be the plane supporting a face of the polyhedron (face f_0 in (7)), leading to somewhat arbitrary absolute values of the type-A functions in (18).

4 *Scene where the performance of the implemented in-*
rence detection algorithm has been tested.

y chosing Π_0 to be the plane containing e_0 that max-
es the distance to the closest vertex of face f_1, the
ved lower bound would be well-behaved in the sense
e. Depending on the trade-off between the cost of
puting this plane and the accuracy needed for the
r bound, the plane can be computed always or only
n the value of the lower bound is below a given
shold distance. There are also procedures for ap-
imating this plane instead of computing it exactly.
ne obtained bounds lead to collision-free translation
nds. If the moving object is required to rotate, and
rotation is expressed with respect to an axis of rota-
the rotation bound can be derived using the trans-
n bound on the vertex of the moving object most
nt from the axis.

7 Implementation

described algorithm has been implemented within
frame of the robot simulator presented in [23], not
for interference detection, but also to *pick out* ge-
tric elements from the screen. It currently runs on
icon Graphics workstation using GL as graphics li-
y. In this library (like in most others, such as CORE
PHIGS), a face is a contour in 3D filled with a given
ur. Thus, a face with holes must be described as a
e contour by adding as many edges as needed. Note
the above algorithm also works in these situations
without any change.

Space limitations prevent us from elaborating on the different prunning strategies we have developed to cut down the computing time when the simplicity of the situation permits. Just to provide a flavor of what these strategies are, let us mention that they belong to three classes: logical, functional and geometrical. Logical prunning is to stop evaluating an AND predicate (alternatively, an OR predicate) when one of the clauses is false (altern., true). Functional prunning is an adaptation of $\alpha - \beta$ prunning to the particular type of XOR graphs we have. Finally, the most particularized type of prunning is that relying on geometric constraints. An example is the extension of the applicability contraints developed in [10] for convex polyhedra, to some non-convex situations.

Figure 4 displays a scene where a manipulation task, involving all the workpieces shown (which in total contain 2388 edges), is simulated. All workpieces are enclosed within spheres to reduce computational overhead when trying to detect interferences. The simulation of the whole task takes 16.95 seconds of CPU. 11.19 % of this time, that is 1.90 seconds, is devoted to detect possible interferences. In the actual implementation this involves 102407 calls to the routine that detects intersections between an edge and a face (corresponding to our key predicate (8)). This routine is executed in an average time of approximately 20 μs.

8 Conclusions and further research

In CAD-CAM and off-line robotic applications requiring high precision, the need to detect interferences between objects as accurately as possible often arises. Then, *efficient* algorithms to do so result in a substantial reduction in the overall time required by such applications.

Since non-convex polyhedra are often used to model industrial workpieces, the aim of this work has been to speed up the detection of interferences between such polyhedra. The speed-up is three-fold: 1) No auxiliary geometric entities are computed, contrarily to the usual way of decomposing the polyhedra (or their faces) into convex entities. 2) The number of additions and multiplications performed is kept as small as possible by considering cross products as primitives and repeatedly using them in subsequent calculations (e.g. no product of two coordinates is computed twice). 3) Prunning strategies can be readily incorporated into the algorithm to avoid superfluous interference tests when the scene allows to do so.

The speed of the algorithm has not only been proved theoretically, but also tested experimentally. In our

ementation, the collisions are indicated (through king) on-line as the user moves a polyhedron on the n using the mouse, a facility of great interest to) users.

ther nice features of the algorithm are its simplicity as been proven to be easy to program), its robust- (since degenerate situations are handled naturally g the same formalism) and the possibility to derive nce bounds directly from it. Moreover, since inter- ce detection is reduced to the additive combination eterminants involving only vertices of the polyhe- collision prevention along predetermined trajecto- can be easily performed.

he next step in our work will be the search for a ersing strategy of *convex* polyhedra that permits re- ng the complexity of the algorithm to $O(n+m)$, h is known to be the optimal one for this particular If this could be achieved, the theoretical interest e algorithm would certainly rise.

nally, let us mention that the mathematical frame- in which the interference detection problem has set admits an easy reformulation in terms of signed coids [1]. This is clearly a theoretical point that de- s further research.

ferences

N. Alon, "The Number of Polytopes, Configurations and Real Matroids," *Mathematika*, Vol. 33, pp. 62-71, 1986.

N. Ahuja, R.T. Chien, R. Yen, and N. Birdwell, "Interference Detection and Collision Avoidance Among Three- Dimensional Objects," in *Proc. 1st Annu. Nat. Conf. Artificial Intel.*, Palo Alto, CA, pp. 44-48, 1980.

J.W. Boyse, "Interference Detection Among Solids and Surfaces," *Com. ACM*, Vol. 21, No. 1, pp. 3-9, January 1979.

J.E. Bobrow, "A Direct Mimimization Approach for Obtaining the Distance between Convex Polyhedra," *The Int. Journal of Robotics Research*, Vol. 8, No. 3, June 1989.

S. Cameron, "A Study of the Clash Detection Problem in Robotics," *Proc. IEEE Conf. on Robotics and Automation*, 1985.

S. Cameron, "Efficient Bounds in Constructive Solid Geometry," *IEEE Computer Graphics and Applications*, pp. 68-74, May 1991.

J.F. Canny, *The Complexity of Robot Motion Planning*, MIT Press., Mass., 1987.

R.K. Culley and K.G. Kempf, "A Collision Detection Algorithm Based on Velocity and Distance Bounds," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1064-1069, 1986.

D.P. Dobkin and D.G. Kirkpatrick, "A Linear Algorithm for Determining the Separation of Convex Polyhedra," *J. Algorithms*, Vol. 6, pp. 381-392, 1985.

B.R. Donald, "Local and Global Techniques for Motion Planning," *Master Thesis*, M.I.T., May 1984.

[11] E.G. Gilbert, D.W. Johnson, "Distance Functions and their Application to Robot Path Planning in the Presence of Obstacles," *IEEE Journal of Robotics and Automation*, Vol. 1, pp. 21-30, 1985.

[12] E.G. Gilbert, D.W. Johnson and S.S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, April 1988.

[13] J.E. Hopcroft, J.T. Schwartz and M. Sharir, "Efficient Detection of Intersections Among Spheres," *Tech. Report* No 59, New York University, February, 1983.

[14] Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The Int. Journal of Robotics Research*, Vol. 5, pp. 90-98, Spring 1986.

[15] D.T. Lee and F.P. Preparata, "Computational Geometry. A Survey," *IEEE Transactions on Computers*, Vol. C-33, No. 12, December 1984.

[16] M.C. Lin and J.F. Canny, "A Fast Algorithm for Incremental Distance Calculation," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1008-1014, 1991.

[17] Y. Liu, S. Arimoto, and H. Noborio, "A New Solid Model HSM and Its Application to Interference Detection between Moving Objects," *Journal of Robotic Systems*, Vol. 8, No. 1, pp. 39-54, 1991.

[18] W. Meyer, "Distances between Boxes: Applications to Collision Detection and Clipping," *Proc. IEEE Int. Conf. on Robotics and Automation*, 1986.

[19] J.R. Rossignac and H.B. Voelcker, "Active Zones in Constructive Solid Geometry for Redundancy and Interference Detection," *IBM Research Report*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, 1986.

[20] J. O'Rourke and N. Badler, "Decomposition of Three-Dimensional Objects into Spheres," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-I, No. 3, July 1979.

[21] E. Rimon and S.P. Boyd, "Efficient Distance Computation Using Best Ellipsoid Fit," *submitted for publication.*

[22] B. Schachter, "Decomposition of Polygons into Convex Sets," *IEEE Trans. on Computers*, Vol. C-27, No. 11, November 1978.

[23] F. Thomas and L. Basañez, "Robot Task Programming through Simulation with Planning Capability," *Preprints of the CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy, September, 1992.

[24] F. Thomas and C. Torras, "Inferring Feasible Assemblies from Spatial Constraints," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 2, pp. 228-239, April 1992.

[25] J. Tornero, J. Hamlin and R.B. Kelley, "Spherical-Object Representation and Fast Distance Computation for Robotic Applications," *IEEE Conf. on Robotics and Automation*, Sacramento, CA, April 1991.

[26] S. Zeghloul, P. Rambeaud and J.P. Lallemand, "A Fast Distance Calculation Between Convex Objects by Optimization Approach," *IEEE Conf. on Robotics and Automation*, pp. 2520-2525, Sacramento, CA, April 1991.