*Journal of Global Optimization* **26:** 361–385, 2003.
© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

# Minimum Distance Between the Faces of Two Convex Polyhedra: A Sufficient Condition

B. LLANAS, M. FERNANDEZ DE SEVILLA and V. FELIU
*Dpto. de Matemática e Informática, E.T.S.I. de Caminos, Ciudad Universitaria, 28040-Madrid, Spain and Dpto. de Ingeniería Eléctrica, E.T.S.I Industriales, Campus Universitario s/n , 13071-Ciudad Real, Spain*

**Abstract.** The problem of finding the Euclidean distance between two convex polyhedra can be reduced to the combinatorial optimization problem of finding the minimum distance between their faces. This paper presents a global optimality criterion for this problem. An algorithm (QLDPA) for the fast computation of the distance between convex and bounded polyhedra is proposed as an application of it. Computer experiments show its fast performance, especially when the total number of vertices is large.

**Key words:** Distance between convex polyhedra, Local descent, Projection algorithms, Sufficient condition of global minimum

## 1. Introduction

The calculation of the distance between bodies has great importance in many applications: collision avoidance in Robotics (Halperin et al., 1997; Khatib, 1986), Virtual Reality simulations (Vince, 1995), and many more.

Some widely used programs in Virtual Reality (VRML, Superscape, etc.) represent solids by means of polyhedra. Therefore, they need to include procedures to calculate the distance between polyhedra in real time.

Realistic modelization of solids requires the use of polyhedra with many vertices, but to our knowledge, data for large polyhedra (more than 350 vertices each) have not been reported.

Let $\mathcal{P}$ and $\mathcal{Q}$ be two convex polyhedra in $R^3$. The problem is to find a point $\mathbf{a} \in \mathcal{P}$ and another point $\mathbf{b} \in \mathcal{Q}$ such that

$$d(\mathbf{a}, \mathbf{b}) = \min_{\substack{\mathbf{x} \in \mathcal{P} \\ \mathbf{y} \in \mathcal{Q}}} d(\mathbf{x}, \mathbf{y}) \equiv d(\mathcal{P}, \mathcal{Q}) \tag{1}$$

A good algorithm for solving (1) should have the following features

1. Find the exact solution or a good approximation.
2. Short computation time.
3. Slow growth of the CPU time versus the complexity of the polyhedra involved.

4. Moderate dependence of the CPU time on the relative position of the polyhedra.

Several methods for solving (1) have been proposed

## 1.1. ALGORITHMS BASED ON STANDARD OPTIMIZATION METHODS

(1) is a quadratic minimization problem subjected to linear constraints and can be solved by conventional optimization methods.

An interior penalty function method with a sequence of unconstrained minimizations would accomplish the goal, but it would also require a large number of iterations to achieve an accurate solution. A quadratic programming method (Lemke's method) has been reported in (Hurteau and Stewart, 1988), but its computation time seems to be long.

An approach based on Rosen's gradient projection method has been studied in Bobrow (1989). In Zeghloul et al. (1992) it is proved that the above method can generate a zigzaging phenomenon when the Kuhn-Tucker conditions are not satisfied for both objects. These critical situations increase the number of iterations and can subsequently curb the convergence of the process. To cope with this phenomenon they propose an algorithm based on a new optimal search direction. This algorithm is better than Bobrow's but its computation time is long in comparison with other methods.

Therefore, the studied classical optimization algorithms do not satisfactorily verify features 1 and 2.

## 1.2. ALGORITHMS THAT REDUCE (1) TO THE PROBLEM OF FINDING THE MINIMUM EUCLIDEAN NORM OF THE POINTS OF A POLYTOPE

Let $P$ and $Q$ be a given pair of finite sets of points of $R^n$. Let us denote the convex hull of $P$ by $C(P)$ and the Minkowski difference of $P$ and $Q$ by

$$P - Q = \{\mathbf{p} - \mathbf{q} \;/\; \mathbf{p} \in P, \quad \mathbf{p} \in Q\}$$

We have

$$d(C(P), C(Q)) = d(\mathbf{0}, C(P - Q)) \tag{2}$$

A basic optimality criterion and an algorithm for solving the right hand side of (2) can be found in Wolfe (1976). A dual algorithm for solving the same problem is described in Fujishinge and Zhan (1990). In Fujishinge and Zhan (1992) the above algorithm is extended for finding the nearest pair of points.

A recursive algorithm, based on Wolfe's criterion, for finding a pair of closest points in two polytopes has been proposed in Sekitani and Yamamoto (1993). The CPU time of a nonrecursive version of the Sekitani-Yamamoto algorithm grows fast with the total number of vertices (Llanas et al., 2000).

An algorithm (GJK) for finding the distance between two polytopes, which is based on the concept of support function and (2), has been proposed in Gilbert et al. (1988). The CPU-time behavior of GJK, although linear in the total number of vertices, presents a rather fast growth.

## 1.3. GEOMETRIC ALGORITHMS

In Dobkin and Kirkpatrick (1990) a hierarchical collection of polytopes, each contained in its predecessor, is used to represent polyhedra. This representation allows for efficient computation of minimum distance. Numerical experiments on this algorithm have not been reported. In Red (1983) an exhaustive search of the distance between all the pairs of bounded planar facets of two polyhedra is used to obtain the minimum distance. The expected running time of this algorithm is $O(N_1.N_2)$ ($N_1$ and $N_2$ being the number of faces of each polyhedron respectively), consequently, this procedure is not suitable for computing the distance between large polyhedra (Llanas et al., 2000).

In Lin (1993) a different approach is described: The Lin-Canny (LC) algorithm. This method starts with a candidate pair of features (vertex, edge or face), one from each polyhedron and check whether the closest points lie on the features. This is done by means of the ‘Applicability Criteria’ which are based on the Voronoi region of each feature. When a pair of features fail the test, the new pair choosen is guaranteed to be closer than the older one. The algorithm must end in a number of steps which are at least equal to the number of feature pairs. The CPU time of this algorithm is linear in the total number of vertices but it appears to grow relatively fast.

## 1.4. OTHER ALGORITHMS

In Cameron (1997) a modification of GJK algorithm, which is based on a technique for minimizing a homogeneous linear function on the vertices of a polyhedron called ‘hill climbing’, is described. The resulting algorithm (GJKC) is very fast but there is a lack of data for large polyhedra.

In Llanas et al. (2000) another procedure, the ‘swap’ algorithm (SA), is proposed. In this method we consider the operator $\pi_{\mathcal{P}\mathcal{Q}} \equiv \pi_{\mathcal{P}} \circ \pi_{\mathcal{Q}}$ where $\pi_{\mathcal{P}}$ and $\pi_{\mathcal{Q}}$ represent the projection operators onto the polyhedra $\mathcal{P}$ and $\mathcal{Q}$, respectively. Since $\pi_{\mathcal{P}\mathcal{Q}}$ is nonexpansive, the theory of nonexpansive operators in Hilbert spaces states that

- $\pi_{\mathcal{P}\mathcal{Q}}$ has at least a fixed point $\mathbf{a}$. This implies that $\mathbf{a}$ and $\pi_{\mathcal{Q}}(\mathbf{a}) = \mathbf{b}$ are a nearest pair of points in $\mathcal{P}$ and $\mathcal{Q}$ (the number of these pairs can be infinite in some cases).

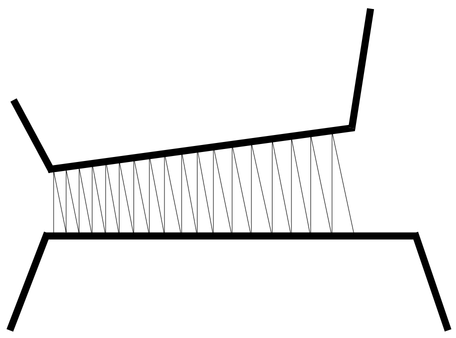

*Figure 1.* Case of slow convergence of SA.

- Point **a** can be computed from an initial point $\mathbf{x}_1 \in \mathcal{P}$ by means of the following iterative process

$$\mathbf{x}_{n+1} = t\ \mathbf{x}_n + (1-t)\ \pi_{\mathcal{P}\mathcal{Q}}(\mathbf{x}_n) \quad t \in (0,1)$$

The projection onto a polyhedron $\pi_{\mathcal{P}}$ is quickly computed using the local search method (LSABF) introduced in Llanas et al (2000). This method obtains the projected point, in the case of 'quasi-spherical' polyhedra, with an expected running time $O(\sqrt{N})$ ($N$ being the number of vertices of the polyhedron). Another procedure of projection (more general but less efficient) can be seen in Llanas and Moreno (1996).

Although SA has shown a fast convergence in a set of computer experiments (the convergence is achieved in only two iterations, that is, four projections point-polyhedron), we have also found that the convergence can be slow in the case of nearby polyhedra with quasiparallel nearest parts of their respective boundaries, if high precision is required (Figure 1).

In this paper, we discretize (1) according to Red's method using the distance between faces instead of between points. If $F(\mathcal{P})$ and $F(\mathcal{Q})$ denote the set of faces of $\mathcal{P}$ and $\mathcal{Q}$ respectively, the distance between the polyhedra can be calculated by

$$d(\mathcal{P},\mathcal{Q}) = \min_{\substack{A\in F(\mathcal{P})\\ B\in F(\mathcal{Q})}} d(A,B) \tag{3}$$

Note that (3) is not valid if a polyhedron is completely contained in the other. This must be kept in mind in the design of any algorithm based on this formula.

We give a global minimum criterion for the distance between two faces to be the minimum in (3). Based on this sufficient (but not necessary) condition we propose a new method for solving (1): the quasilocal descent-projection algorithm (QLDPA). This algorithm reduces the problem (3) to the computation of a comparatively little number of distances between polygons in $R^3$. QLDPA overcomes the problems of SA and exhibits a slow growth of CPU time versus the total number of vertices.

This paper is organized as follows: Section 2 provides the global optimality result, Section 3 describes the new algorithm and gives a proof of its convergence,

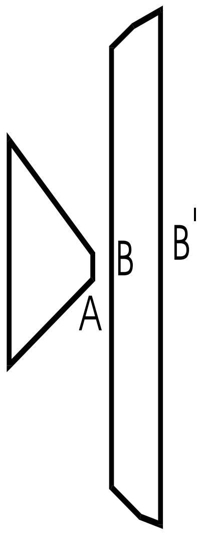

*Figure 2.* Minima of $d$.

Section 4 details numerical experiments and discussion of the results and Section 5 provides some concluding remarks.

## 2. A sufficient condition for the distance between the faces of two convex polyhedra to be minimum

First, we emphasize the difference between local and global minima of the function which appears at the right side of (3), by means of an example in $R^2$.

The global minimum is reached at the faces $(A, B)$, but a local minimum at faces $(A, B')$ exists (Figure 2).

If we consider the fact that the local minimum is reached at a pair of *invisible faces* while the global minimum is at a pair of *reciprocally visible faces* $(RV)$ we have a clue for finding the cited sufficient condition.

We will now state the above intuitive concepts more precisely. From now on, $R^3$ will denote the Euclidean 3-space. The closed and half-open segments in $R^3$ are defined by

$$[\mathbf{a}, \mathbf{b}] \equiv \{\mathbf{x} \in R^3/\mathbf{x} = \mathbf{a} + t(\mathbf{b} - \mathbf{a}), \quad t \in [0, 1]\}$$

$$]\mathbf{a}, \mathbf{b}] \equiv \{\mathbf{x} \in R^3/\mathbf{x} = \mathbf{a} + t(\mathbf{b} - \mathbf{a}), \quad t \in (0, 1]\}$$

$d$ denotes the Euclidean distance, "." denotes the inner product in $R^3$, and $|\mathbf{v}|$ the Euclidean norm of $\mathbf{v}$.

A face $A$ of a polyhedron $\mathcal{P}$ can be considered as an oriented polygon, that is, a polygon with a normal vector $\mathbf{n}_A$ oriented toward the exterior of $\mathcal{P}$. We represent it by $(A, \mathbf{n}_A)$ and denote by $F(\mathcal{P})$ the set of faces of $\mathcal{P}$.

- Let $\mathcal{P}$ be a polyhedron and $\mathbf{b} \in R^3$. We say that the face $(A, \mathbf{n}_A)$ *is visible from* $\mathbf{b}$ if for any $\mathbf{a} \in A$ we have

  $$(\mathbf{a} - \mathbf{b}).\mathbf{n}_A \leqslant 0$$

  The set of faces of $\mathcal{P}$ visible from $\mathbf{b}$ will be denoted by $Vis_{\mathcal{P}}(\mathbf{b})$.
- If $(A, \mathbf{n}_A) \in F(\mathcal{P})$ and $(B, \mathbf{n}_B) \in F(\mathcal{Q})$ we say that they are *reciprocally visible* if there exists a nearest pair of points in the two faces $\mathbf{a} \in A$ and $\mathbf{b} \in B$ such that $(A, \mathbf{n}_A)$ is visible from $\mathbf{b}$ and $(B, \mathbf{n}_B)$ is visible from $\mathbf{a}$. We will denote this relation by $(A, B) \in RV$.

The first definition is obviously independent of the point $\mathbf{a} \in A$. The second definition is also independent of the nearest pair of points $\mathbf{a}$ and $\mathbf{b}$ as is proven in the following Lemma.

LEMMA 1. *Let $A$ and $B$ be two closed convex sets in $R^3$, such that, $A \cap B = \emptyset$. If $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{a}', \mathbf{b}')$ are nearest pairs of points in $A$ and $B$ respectively, then*

$$\mathbf{a} - \mathbf{b} = \mathbf{a}' - \mathbf{b}'$$

*Proof.* Since $\mathbf{b}$ is the projection of $\mathbf{a}$ onto $B$ (Balakrishnan, 1981, p. 10)

$$(\mathbf{b} - \mathbf{a}).(\mathbf{q} - \mathbf{b}) \geqslant 0 \quad \forall \mathbf{q} \in B \tag{4}$$

Since $\mathbf{a}$ is the projection of $\mathbf{b}$ onto $A$

$$(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{p}) \geqslant 0 \quad \forall \mathbf{p} \in A \tag{5}$$

Since $\mathbf{b}' - \mathbf{a}' = \mathbf{b}' - \mathbf{b} + \mathbf{b} - \mathbf{a} + \mathbf{a} - \mathbf{a}'$, we have

$$|\mathbf{b}' - \mathbf{a}'|^2 = |\mathbf{b} - \mathbf{a}|^2 + |\mathbf{b}' - \mathbf{b} + \mathbf{a} - \mathbf{a}'|^2 + 2(\mathbf{b} - \mathbf{a}).(\mathbf{b}' - \mathbf{b}) + 2(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{a}')$$

By (4) and (5) the last two addends are not negative. Thus, since $|\mathbf{b}' - \mathbf{a}'| = |\mathbf{b} - \mathbf{a}|$, we have

$$|\mathbf{b}' - \mathbf{b} + \mathbf{a} - \mathbf{a}'| = 0$$

The result follows. □

We will use the following lemmas further on.

LEMMA 2. *Let $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in R^3$ be such that $|\mathbf{d} - \mathbf{c}| < |\mathbf{b} - \mathbf{a}|$. Then at least one of the quantities $(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b})$ and $(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{c})$ is negative.*

*Proof.*

$$\begin{aligned} |\mathbf{d} - \mathbf{c}|^2 &= |(\mathbf{b} - \mathbf{a}) + (\mathbf{a} - \mathbf{c} + \mathbf{d} - \mathbf{b})|^2 = \\ &= |\mathbf{b} - \mathbf{a}|^2 + |\mathbf{a} - \mathbf{c} + \mathbf{d} - \mathbf{b}|^2 + 2(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{c} + \mathbf{d} - \mathbf{b}) \end{aligned}$$

Hence

$$0 > |\mathbf{d} - \mathbf{c}|^2 - |\mathbf{b} - \mathbf{a}|^2 \geqslant 2(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{c}) + 2(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b})$$

The result follows. □

LEMMA 3. *Let* $\mathbf{a}, \mathbf{b}, \mathbf{d} \in R^3$ *be such that* $(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b}) < 0$. *For* $t \in (0, 1]$ *let us define* $\mathbf{b}_t = \mathbf{b} + t(\mathbf{d} - \mathbf{b})$. *Then* $d(\mathbf{a}, \mathbf{b}_t) < d(\mathbf{a}, \mathbf{b})$ *for* $t > 0$ *small enough.*

*Proof.*

$$\begin{aligned} |\mathbf{b}_t - \mathbf{a}|^2 &= |\mathbf{b} - \mathbf{a} + t(\mathbf{d} - \mathbf{b})|^2 \\ &= |\mathbf{b} - \mathbf{a}|^2 + 2t(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b}) + t^2|\mathbf{d} - \mathbf{b}|^2 \end{aligned}$$

The result follows. □

The next result will establish that if a local minimum of the function *distance between faces* is reached at a pair of reciprocally visible faces, then it is the global minimum.

Previous to the proof of this statement, we need to introduce some definitions and an auxiliary function studied in Llanas et al. (2000).

From now on, if $A \in F(\mathcal{P})$, we denote by $N(A)$ the set of faces of $\mathcal{P}$ which have a point or an edge in common with $A$ excluding from this set $A$ itself. We denote by $NA$ the number of elements of $N(A)$.

Let $\mathbf{b}$ be an external point to $\mathcal{P}$ and $\mathbf{x}$ a point belonging to $\mathcal{P}$. We denote by $\partial\mathcal{P}$ the boundary of $\mathcal{P}$. Using the notation $I = [\mathbf{x}, \mathbf{b}] \cap \partial\mathcal{P}$ we define the auxiliary function $f$ as follows

$$\begin{aligned} f : \mathcal{P} &\rightarrow \partial\mathcal{P} \\ \mathbf{x} &\rightarrow \{\mathbf{z}/\mathbf{z} \in I \ and \ |\mathbf{z} - \mathbf{b}| \leqslant |\mathbf{y} - \mathbf{b}| \quad \forall \mathbf{y} \in I\} \end{aligned}$$

The function $f$ has the following properties (Llanas et al., 2000)

1. $f$ is continuous on $\mathcal{P}$.
2. If $\mathbf{x} \in Vis_{\mathcal{P}}(\mathbf{b})$ then $f(\mathbf{x}) = \mathbf{x}$
3. For all $\mathbf{x} \in \mathcal{P}$ we have that $f(\mathbf{x}) \in Vis_{\mathcal{P}}(\mathbf{b})$
4. If $\mathbf{x} \in \mathcal{P}$ and $\mathbf{x} \notin Vis_{\mathcal{P}}(\mathbf{b})$ then $f(\mathbf{x}) \neq \mathbf{x}$

LEMMA 4. *Let* $\mathcal{P}$ *and* $\mathcal{Q}$ *be two convex polyhedra. Let* $A \in F(\mathcal{P})$ *and* $B \in F(\mathcal{Q})$ *be such that* $(A, B) \in RV$. *If* $A' \in F(\mathcal{P})$ *and* $B' \in F(\mathcal{Q})$ *such that* $d(A', B') < d(A, B)$ *then there exists* $\tilde{B} \in N(B)$ *verifying* $d(A, \tilde{B}) < d(A, B)$ *or there exists* $\tilde{A} \in N(A)$ *verifying* $d(\tilde{A}, B) < d(A, B)$ *or both.*

*Proof.* Let $\mathbf{a} \in A$ and $\mathbf{b} \in B$ be a nearest pair of points of the faces $A$ and $B$, and $\mathbf{c} \in A'$ and $\mathbf{d} \in B'$ a nearest pair of points of the faces $A'$ and $B'$. We have by hypothesis that $d(\mathbf{c}, \mathbf{d}) < d(\mathbf{a}, \mathbf{b})$. By Lemma 2 we have that only one of the following posibilities can arise

- $(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b}) < 0$ and $(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{c}) \geqslant 0$
- $(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b}) \geqslant 0$ and $(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{c}) < 0$
- $(\mathbf{b} - \mathbf{a}).(\mathbf{d} - \mathbf{b}) < 0$ and $(\mathbf{b} - \mathbf{a}).(\mathbf{a} - \mathbf{c}) < 0$

We only give the proof in the first case, the others are similar.

Since $(\mathbf{b}-\mathbf{a}).(\mathbf{d}-\mathbf{b}) < 0$, Lemma 3 states that there exists $\mathbf{b}' \in ]\mathbf{b}, \mathbf{d}]$ arbitrarily near to $\mathbf{b}$ and satisfying

$$d(\mathbf{a}, \mathbf{b}') < d(\mathbf{a}, \mathbf{b}) \tag{6}$$

From the continuity of the auxiliary function $f$ referred to the point $\mathbf{a}$ and the polyhedron $\mathcal{Q}$, we have that given $\epsilon > 0$, there exists $\delta > 0$ such that

$$f[B_\delta(\mathbf{b}) \cap \mathcal{Q}] \subset B_\epsilon(f[\mathbf{b}]) \cap \partial\mathcal{Q}$$

where $B_\gamma(\mathbf{c})$ is the open ball of radius $\gamma$ centered in $\mathbf{c}$. Since $\mathbf{b}$ belongs to a visible face from $\mathbf{a}$ we have that $f(\mathbf{b}) = \mathbf{b}$ and therefore

$$f[B_\delta(\mathbf{b}) \cap \mathcal{Q}] \subset B_\epsilon(\mathbf{b}) \cap \partial\mathcal{Q}$$

If we take a point $\mathbf{b}' \in B_\delta(\mathbf{b}) \cap ]\mathbf{b}, \mathbf{d}]$ ($]\mathbf{b}, \mathbf{d}]$ is contained in $\mathcal{Q}$ by convexity) that satisfies (6) then we have that $f(\mathbf{b}') \in B_\epsilon(\mathbf{b}) \cap \partial\mathcal{Q}$, and

$$d(\mathbf{a}, f(\mathbf{b}')) \leqslant d(\mathbf{a}, \mathbf{b}') < d(\mathbf{a}, \mathbf{b})$$

Therefore a point exists on the boundary $\partial\mathcal{Q}$ arbitrarily close to the face $B$ and placed on a face $\tilde{B}$ visible from $\mathbf{a}$, that is, $\tilde{B} \in N(B)$, $\tilde{B} \in Vis_{\mathcal{Q}}(\mathbf{a})$ and $d(A, \tilde{B}) < d(A, B)$. □

The above result can be stated in the following equivalent form, that constitutes the **sufficient condition of global minimum**.

THEOREM 1. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two convex polyhedra. Let $A \in F(\mathcal{P})$ and $B \in F(\mathcal{Q})$ be such that $(A, B) \in RV$. If*
1. *$d(\tilde{A}, B) \geqslant d(A, B)$ for all $\tilde{A} \in N(A)$;*
2. *$d(A, \tilde{B}) \geqslant d(A, B)$ for all $\tilde{B} \in N(B)$;*

*then*

$$d(A, B) \leqslant d(A', B') \text{ for all } A' \in F(\mathcal{P}) \text{ and for all } B' \in F(\mathcal{Q}).$$

Hypotheses 1 and 2 are, in fact, the formal definition of local minimum of the distance between faces. The existence of at least a pair of faces verifying these hypotheses can be proven using the results presented in Llanas et al. (2000).

## 3. The quasilocal descent-projection algorithm

In this section we describe a new algorithm for finding the distance between two convex and bounded polyhedra: *The quasilocal descent-projection algorithm* (QLDPA). The polyhedra must be represented by the data of the faces (Section 4.1).

The main idea of QLDPA consists of performing a local descent from an arbitrary pair of reciprocally visible faces $(A_0, B_0)$ to a “neighbouring” pair $(A_0, \tilde{B}) \in RV$ or $(\tilde{A}, B_0) \in RV$ in such a way that the distance between the faces decreases. The process is repeated until reaching a local minimum which is, by Theorem 1, the global one.

This naive algorithm is not feasible due to the following facts

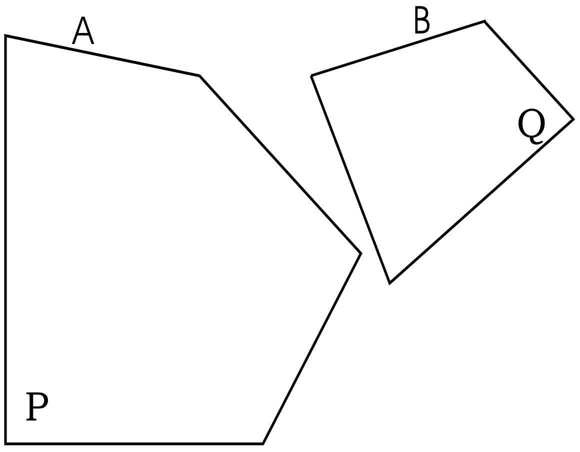

*Figure 3.* Faces $A$ and $B$ have the $WVD$ property.

- It seems difficult to find two initial reciprocally visible faces without an exhaustive search.
- In general, there exist pairs of faces $(A, B)$ verifying
  - $*$ $(A, B) \in RV$.
  - $*$ The global minimum of the distance between faces is not attained at $(A, B)$.
  - $*$ If $\tilde{A} \in N(A)$ is such that $d(\tilde{A}, B) < d(A, B)$ then $(\tilde{A}, B) \notin RV$ and if $\tilde{B} \in N(B)$ is such that $d(A, \tilde{B}) < d(A, B)$ then $(A, \tilde{B}) \notin RV$.

  We denote such a pair of faces without the possibility of visible descent by $(A, B) \in WVD$. Figure 3 shows an example in the plane. The above algorithm would stop if it arrived at a $WVD$ pair of faces.

For avoiding this difficulty we modify the algorithm in the following way (Projection Step): If a pair $(A, B) \in WVD$ is reached in the descent process, and $\mathbf{a} \in A$ and $\mathbf{b} \in B$ are such that $d(\mathbf{a}, \mathbf{b}) = d(A, B)$, we compute $\mathbf{bp} = \pi_{\mathcal{Q}}(\mathbf{a})$ (projection of $\mathbf{a}$ on $\mathcal{Q}$) and then $\mathbf{ap} = \pi_{\mathcal{P}}(\mathbf{bp})$ (projection of $\mathbf{bp}$ on $\mathcal{P}$). We determine a face $B_j$ such that $\mathbf{bp} \in B_j$. If the test of convergence of SA fails and $(A, B_j) \in RV$ we apply a local descent step. Otherwise we project again according to the methodology of SA, but keeping in mind the $RV$ property of the faces that contain the projected points. When a subsequent pair of faces is reciprocally visible, we apply a local descent step and so on.

In this way, we have a global minimum convergence test after a little number of steps of the SA avoiding unnecessary projections when the faces reached are a pair of nearest ones. On the other hand this procedure allows the stop of the algorithm when it arrives at a $WVD$ pair of faces to be avoided.

Figure 4 shows a simplified flowchart of QLDPA.

If local (visible or invisible) descent cannot be performed from a pair of $RV$ faces, the sufficient condition of global minimum (SCGM) is applied.

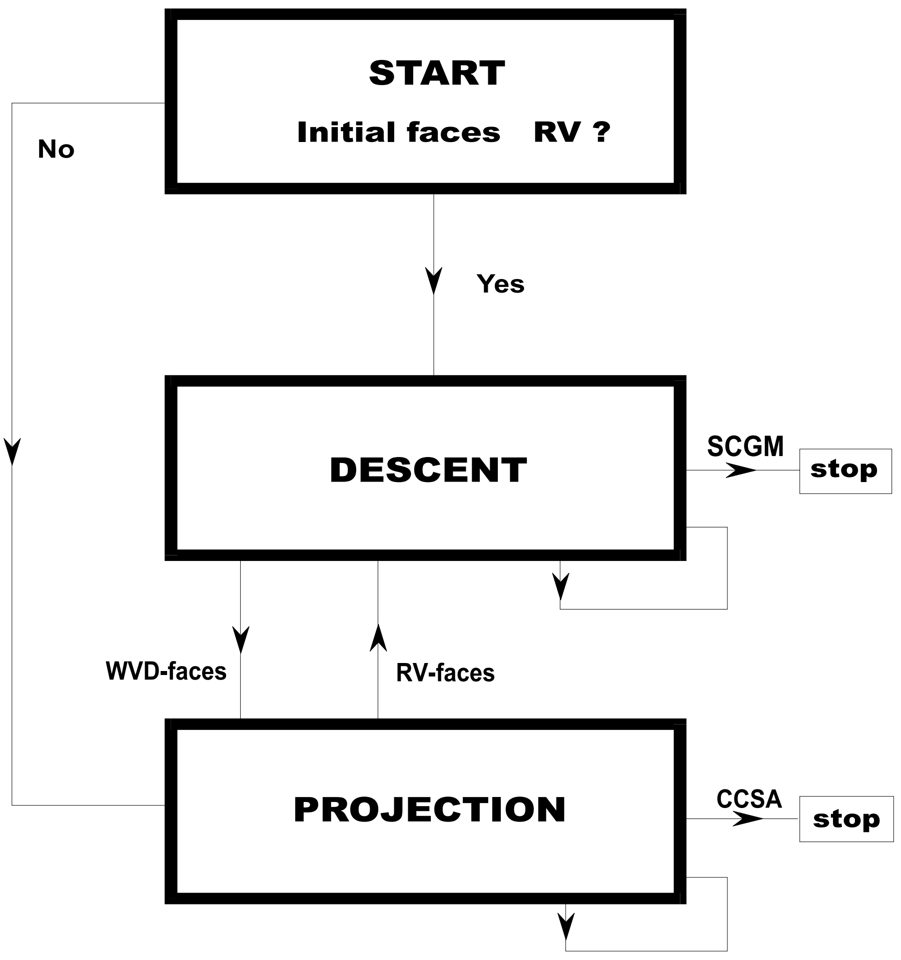

*Figure 4.* General flowchart of QLDPA.

After a projection step, the convergence condition of the ‘swap’ algorithm (CCSA) is applied.

### 3.1. QUASILOCAL DESCENT-PROJECTION ALGORITHM (QLDPA)

Below we give a FORTRAN 90-like pseudocode of QLDPA. We assume that the data of both polyhedra, the precision $EPS$ and the parameter $t$ have been read. We will use the following notation

$$FP(\mathbf{x}) = \{A_i \in F(\mathcal{P})/\mathbf{x} \in A_i\}$$

When $\mathbf{x}$ is the projection of a point onto $\mathcal{P}$, $FP(\mathbf{x})$ can be computed by the projection algorithm LSABF (Llanas et al., 2000).

```
                I (Start)

– Choose two faces $A_0 \in F(\mathcal{P})$ and $B_0 \in F(\mathcal{Q})$
at random or with some strategy (Section 4).
– Find a nearest pair of points $\mathbf{a}_0$ and $\mathbf{b}_0$
in the faces $A_0$ and $B_0$.
– Compute the distance between $A_0$ and $B_0$
  $D = d(A_0, B_0) = d(\mathbf{a}_0, \mathbf{b}_0)$
if ($\mathbf{a}_0 = \mathbf{b}_0$) then
  print $d(\mathcal{P}, \mathcal{Q}) = 0$
  STOP
endif
$\mathbf{a} = \mathbf{a}_0$
$A = A_0, B = B_0$
if $[(A, B) \in RV]$ then
  go to II(1)
else
  go to V(1)
endif
```

```
II (Descent)

(1) $VIS = 1$
$d = D$
do $i = 1, \ldots\ldots, NA$ $(A_i \in N(A))$
    - Find a nearest pair
    of points $\mathbf{a}_i$ and $\mathbf{b}_i$
    in the faces $A_i$ and $B$.
    - Compute the distance
    between $A_i$ and $B$
      $d_i = d(A_i, B) = d(\mathbf{a}_i, \mathbf{b}_i)$
      if $(d_i = 0)$ then
        print $d(\mathcal{P}, \mathcal{Q}) = 0$
        STOP
      endif
      if $(d_i < d)$ then
        if $((A_i, B) \in RV)$ then
          $D = d_i$, $A = A_i$
          go to II(1)
        else
          $\mathbf{a} = \mathbf{a}_i$
          $D = d_i$
          $VIS = 0$
        endif
      endif
enddo
go to III
```

```
III (Descent)

do $j = 1, \ldots\ldots, NB$ $(B_j \in N(B))$
    - Find a nearest pair
    of points $\mathbf{a}_j$ and $\mathbf{b}_j$
    in the faces $A$ and $B_j$.
    - Compute the distance
    between $A$ and $B_j$
      $d_j = d(A, B_j) = d(\mathbf{a}_j, \mathbf{b}_j)$
      if $(d_j = 0)$ then
        print $d(\mathcal{P}, \mathcal{Q}) = 0$
        STOP
      endif
      if$(d_j < d)$ then
        if $((A, B_j) \in RV)$ then
          $D = d_j$, $B = B_j$
          go to II(1)
        else
          $\mathbf{a} = \mathbf{a}_j$
          $D = d_j$
          $VIS = 0$
        endif
      endif
enddo
go to IV
```

In II and III, $VIS = 1$ indicates that $(A, B) \in RV$ and the possibility of visible descent must be checked.

```
IV (Descent)

if $(VIS = 1)$ then
    print $d(\mathcal{P}, \mathcal{Q}) = D$
    STOP
else
    go to V (1)
endif
```

```
V (Projection)

- (1) Compute $\mathbf{bp} = \pi_{\mathcal{Q}}(\mathbf{a})$
- Compute $\mathbf{ap} = \pi_{\mathcal{P}}(\mathbf{bp})$
if ($\mathbf{bp} \in \mathcal{P}$, that is, $\mathbf{ap} = \mathbf{bp}$) then
    print $d(\mathcal{P}, \mathcal{Q}) = 0$
    STOP
endif
Find $A_i \in FP(\mathbf{a})$ and $B_j \in FP(\mathbf{bp})$
if $(d(\mathbf{ap}, \mathbf{a}) < EPS)$ then
    print $d(\mathcal{P}, \mathcal{Q}) = d(A_i, B_j)$
    STOP
else if $((A_i, B_j) \in RV)$ then
    $D = d(A_i, B_j) = d(\mathbf{pa}, \mathbf{pb})$
    $A = A_i, B = B_j$
    go to II (1)
else
    $\mathbf{a} = t\mathbf{a} + (1-t)\mathbf{ap}$
    go to V(1)
endif
```

In IV, $VIS = 1$ indicates that the previous pair $(A, B) \in RV$ in II-III is the sought solution and $VIS = 0$ indicates that the previous pair $(A, B)$ in II-III is a $WVD$ one.

For proving the convergence of QLDPA we need the following result

LEMMA 5. *Let $\mathcal{P}$ and $\mathcal{Q}$ be two closed convex sets in $R^n$, let us consider the operator $\pi_{\mathcal{P}\mathcal{Q}} \equiv \pi_{\mathcal{P}} \circ \pi_{\mathcal{Q}}$ where $\pi_{\mathcal{P}}$ and $\pi_{\mathcal{Q}}$ represent the projection operators onto the sets $\mathcal{P}$ and $\mathcal{Q}$, respectively. Let $\mathbf{a}_1$ be an arbitrary point of $\mathcal{P}$. Let us define the sequence*

$$\mathbf{a}_{n+1} = t\mathbf{a}_n + (1-t)\pi_{\mathcal{P}\mathcal{Q}}(\mathbf{a}_n)$$

*If we call $d_{\mathcal{Q}}(\mathbf{a}_n)$ the distance from $\mathbf{a}_n$ to $\mathcal{Q}$, we have*

$$d_{\mathcal{Q}}(\mathbf{a}_{n+1}) \leqslant d_{\mathcal{Q}}(\mathbf{a}_n) \quad for\ all\ n. \tag{7}$$

*Proof.* The distance from a point to a fixed, non-empty and convex set in $R^n$ is a convex function (Rockafellar, 1970, p. 34), therefore

$$d_{\mathcal{Q}}(\mathbf{a}_{n+1}) = d_{\mathcal{Q}}(t\mathbf{a}_n + (1-t)\pi_{\mathcal{P}\mathcal{Q}}(\mathbf{a}_n))$$

$$\leqslant t d_{\mathcal{Q}}(\mathbf{a}_n) + (1-t) d_{\mathcal{Q}}(\pi_{\mathcal{P}\mathcal{Q}}(\mathbf{a}_n)) \tag{8}$$

On the other hand, we have

$$d_{\mathcal{Q}}(\mathbf{a}_n) = |\mathbf{a}_n - \pi_{\mathcal{Q}}(\mathbf{a}_n)| \geqslant |\pi_{\mathcal{P}}(\pi_{\mathcal{Q}}(\mathbf{a}_n)) - \pi_{\mathcal{Q}}(\mathbf{a}_n)|$$

$$\geqslant |\pi_{\mathcal{P}}(\pi_{\mathcal{Q}}(\mathbf{a}_n)) - \pi_{\mathcal{Q}}(\pi_{\mathcal{P}}(\pi_{\mathcal{Q}}(\mathbf{a}_n)))| = d_{\mathcal{Q}}(\pi_{\mathcal{P}\mathcal{Q}}(\mathbf{a}_n)) \tag{9}$$

Inequality (7) follows from (8) and (9). □

THEOREM 2. *QLDPA converges to a nearest pair of points* **a** *and* **b** *in the polyhedra* $\mathcal{P}$ *and* $\mathcal{Q}$*, when* $EPS \to 0$.

*Proof.* The algorithm consists of the repetition (infinite or until achieving convergence) of successive steps of the following types: $D_1$, $D_2$, $P_1$, $P_2$.

*Steps of type D (Descent)*

data: Two $RV$ faces and their distance $D$. We have one of the following possibilities

- The sufficient condition of global minimum is satisfied (convergence).
- By descent we arrive at a pair of $RV$ faces and evaluate their distance $D'$ (Step $D_1$).
- By descent we arrive to a pair of faces which are not reciprocally visible and calculate a nearest pair of points **a** and **b** in them and their distance $D'$ (Step $D_2$).

*Steps of type P (Projection)*

data: A point **a** on the polyhedron $\mathcal{P}$ and the previous distance $D$.

Calculate the double projection $\mathbf{ap} = \pi_{\mathcal{P}}(\pi_{\mathcal{Q}}(\mathbf{a}))$ and the faces containing **a** and **bp**.

Verify the convergence condition $|\mathbf{a} - \mathbf{ap}| < EPS$. Then we have one of the following possibilities

- Convergence.
- If the faces containing **a** and **bp** are reciprocally visible then calculate their distance $D'$ and begin a new step of type D (Step $P_1$).
- If the above pair of faces does not have the $RV$ property, make $\mathbf{a} = t\mathbf{a} + (1-t)\mathbf{ap}$ and begin a new step of type P (Step $P_2$).

From the above definitions we have that the four types of steps must appear in the following order

$$D_1 \to D_1 \ \ or \ \ D_2$$
$$D_2 \to P_1 \ \ or \ \ P_2$$
$$P_1 \to D_1 \ \ or \ \ D_2$$
$$P_2 \to P_1 \ \ or \ \ P_2$$

First, we prove that the sequence of distances between faces generated by the algorithm $\{D\}$ is monotonically decreasing. In effect, steps of type $D_1$ or $D_2$ calculate a new distance $D^{'}$, such that, $D^{'} < D$. Type $P_2$ steps do not evaluate a new distance in an explicit way. From the above stated order we conclude that steps $P_1$ only can appear in one of the following ways

1. $D_2, P_1$
2. $I_0, P_1$
3. $I_0, P_2, P_2, \ldots.(k\ times)\ldots, P_2, P_1$
4. $D_2, P_2, P_2, \ldots.(k\ times)\ldots, P_2, P_1$

where we denote by $I_0$ a pair of arbitrary initial faces $A$ and $B$. We have in the four cases that $D^{'} \leqslant D$. In effect, in cases 1 and 2 we begin with a pair of faces which are separated by a distance $D$ attained at the points **a** and **b**.

We have

$$D^{'} = d(\mathbf{pa}, \mathbf{pb}) \leqslant d(\mathbf{a}, \mathbf{bp}) \leqslant d(\mathbf{a}, \mathbf{b}) = D$$

In cases 3 and 4 we also begin with a pair of faces separated by a distance $D$ attained at the points **a** and **b**.

In the first application of $P_2$, we have

$$d(\mathbf{a}_1, \mathbf{bp}_1) \leqslant d(\mathbf{a}, \mathbf{b}) \quad (here, \quad \mathbf{a}_1 = \mathbf{a})$$

In the steps that follow, several points $\mathbf{a}_n$ on $\mathscr{P}$ are computed by means of the formula

$$\mathbf{a}_{n+1} = t\mathbf{a}_n + (1-t)\pi_{\mathscr{P}\mathscr{Q}}(\mathbf{a}_n)$$

By Lemma 5, the sequence $\{d(\mathbf{a}_n, \mathbf{bp}_n)\}$ is monotonically decreasing.

Therefore, after $k$ steps $P_2$ and one step $P_1$ we have

$$D^{'} = d(\mathbf{pa}, \mathbf{pb}) \leqslant d(\mathbf{a}_{k+1}, \mathbf{bp}_{k+1}) \leqslant \ldots.. \leqslant d(\mathbf{a}_1, \mathbf{bp}_1) \leqslant d(\mathbf{a}, \mathbf{b}) = D$$

And the new distance is less than or equal to the initial one.

The above results imply that $\{D\}$ is monotonically decreasing.

The convergence criterion contained in IV is a direct consequence of Theorem 1. The convergence criterion in V is that corresponding to SA.

The only way of divergence of QLDPA would be the generation of an infinite sequence of steps $D_i, P_j, \ldots$ that do not satisfy any of the previous criteria. Since the sequence $\{D\}$ is monotonically decreasing, the number of steps of type D in the sequence is finite because

- The distance is strictly decreased in every step of type D, this means that every pair of faces is visited only one time.
- The number of pairs of faces is finite.

Since $P_1$ must be followed by a step of type D the same reasoning implies that $P_1$ can only appear a finite number of times.

This means that from a given term onward the algorithm is a succession of steps of type $P_2$. Therefore the algorithm coincides with SA and is convergent as was proven in Llanas et al (2000). □

## 4. Computational Experiments

### 4.1. EXPERIMENTAL METHODOLOGY

To calculate the distance between different types of polyhedra of increasing degree of complexity we have used QLDPA.

QLDPA was programmed using Watcom C, and the computer experiments were performed using a Pentium III (866 Mhz) processor.

As projector onto a polyhedron we used LSABF (Llanas et al., 2000). In this paper we have improved the above implementation of LSABF using a 'hill climbing' method for finding an initial visible face.

Experiments are shown in Figures 5,6 and 10. They are as follows

- Tetrahedron: four vertices and four faces.
- Parallelepiped: eight vertices and six faces.
- Icosahedron: twelve vertices and twenty faces. The considered icosahedra have been obtained by means of rigid motions of the following one

| Vertices of the reference icosahedron | | | | | |
|---|---|---|---|---|---|
| (0,t, 1 ) | (0,-t,-1) | (0, t, -1 ) | (0, -t, 1 ) | (1, 0, t) | (-1, 0, t ) |
| (1,0,-t) | (-1, 0, -t) | (t,1,0) | (-t, -1, 0) | (t,-1,0) | (-t, 1, 0 ) |

Here $t = 1.618034$.

- Polyhedra of type $\mathcal{E}$: This kind of polyhedra are those transformed by a rigid motion of polyhedra inscribed in the ellipsoid

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{(z-c)^2}{c^2} = 1 \tag{10}$$

The ellipsoid has been subdivided axially ($z$ axis) into $m$ parts and radially ($xy$ plane) into $n$ parts. The resulting polyhedron has $(m-2)n+2$ vertices and $(m-1)n$ faces.

- Polyhedron of 'ladder' type: 14 vertices and 9 faces (left side of Figure 10).

Previous to the presentation of the results, some remarks are necessary.

- The choice of the initial faces $A_0 \in F(\mathcal{P})$ and $B_0 \in F(\mathcal{Q})$ of QLDPA has been made by means of the following procedure
  1. Calculate the centers **cp** and **cq** of the bounding boxes of $\mathcal{P}$ and $\mathcal{Q}$ the sides of which are parallel to the coordinate axes (Glaeser, 1994). When

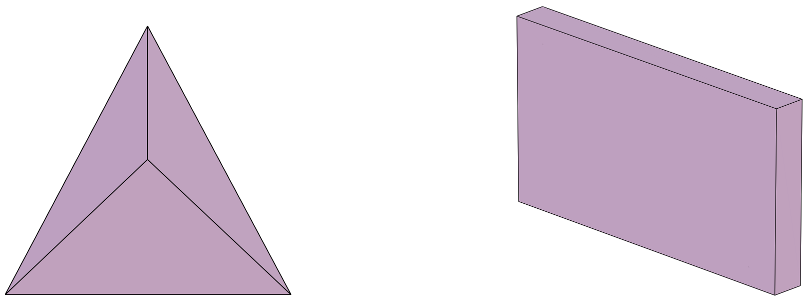

*Figure 5.* Tetrahedron and Parallelepiped.

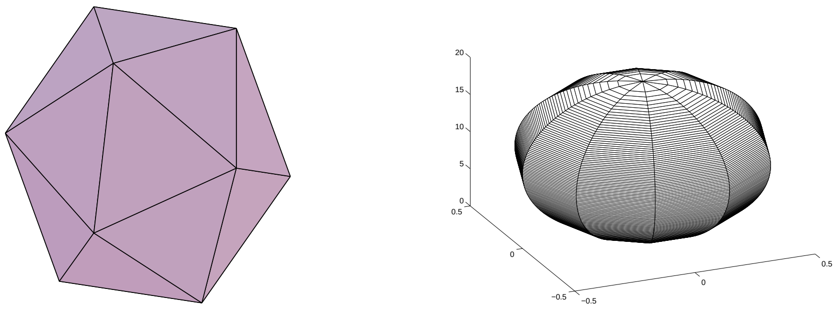

*Figure 6.* Icosahedron and Polyhedron of type $\mathcal{E}$.

the polyhedra are transformed by a rigid motion, **cp** and **cq** are transformed according to it. If **cp** or **cq** are not contained inside the polyhedra, the respective barycenters can be used.
Put $\mathbf{s} = \mathbf{cq} - \mathbf{cp}$.

2. If $|\mathbf{s}| = 0$ then $d(\mathcal{P}, \mathcal{Q})= 0$.
3. Determine a vertex $\mathbf{v}_0 \in V(\mathcal{Q})$ such that

$$\mathbf{v}_0.\mathbf{s} \leqslant \mathbf{v}_j.\mathbf{s} \qquad \forall \mathbf{v}_j \in V(\mathcal{Q})$$

   where $V(\mathcal{Q})$ denotes the set of vertices of $\mathcal{Q}$.
4. Calculate the projection $\mathbf{p} = \pi_{\mathcal{P}}(\mathbf{v}_0)$. Put $\mathbf{r} = \mathbf{p} - \mathbf{v}_0$.
5. If $|\mathbf{r}| = 0$ then $d(\mathcal{P}, \mathcal{Q})= 0$.
6. Determine a face $(PA, \mathbf{n}) \in F(\mathcal{P})$ such that $\mathbf{p} \in PA$ and $\mathbf{r}.\mathbf{n} \leqslant 0$ (this is always feasible (Llanas et al., 2000)).
7. Determine a face $(B, \mathbf{n}_B) \in F(\mathcal{Q})$ such that $\mathbf{v}_0 \in B$ and preferably fulfilling $\mathbf{r}.\mathbf{n}_B \geqslant 0$.
8. Finally, make $A_0 = PA$ and $B_0 = B$

By carrying out the procedure 1–8 we increase the probability of starting the algorithm with two reciprocally visible faces. In any case, the resulting faces are fairly near in comparison to the remaining pairs.

*Table 1.* Low complexity experiments (time $\times 10^{-6}s$)

| N | $m$ | $n$ | $EPS$ | $t$ | PREC | TSA | NISA | TQLDPA | NDP |
|---|---|---|---|---|---|---|---|---|---|
| 8 | - | - | $10^{-6}$ | 0 | $< 10^{-6}$ | 38 | 2 | 93 | 7 |
| 16 | - | - | $10^{-6}$ | 0 | $< 10^{-6}$ | 44 | 2 | 143 | 9 |
| 24 | - | - | $10^{-6}$ | 0 | $< 10^{-6}$ | 124 | 2 | 294 | 19 |
| 44 | 7 | 4 | $10^{-6}$ | 0 | $< 10^{-6}$ | 98 | 2 | 411 | 17 |
| 104 | 7 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 136 | 2 | 418 | 17 |
| 224 | 13 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 137 | 2 | 421 | 17 |
| 304 | 17 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 134 | 2 | 418 | 17 |
| 424 | 23 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 134 | 2 | 421 | 17 |
| 504 | 27 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 142 | 2 | 423 | 17 |
| 704 | 37 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 148 | 2 | 473 | 17 |

- The algorithm needs the following topological data for every polyhedron processed

  * Number of vertices and faces of $\mathcal{P}$.
  * Number and description of the vertices of every face.
  * Number and description of the vertices adjacent to every vertex.
  * Number and description of the faces adjacent to every face.
  * Number and description of the faces intersecting in every vertex.

  It also uses the coordinates of the vertices of each polyhedron.

- Compute time for two given polyhedra in a fixed position is calculated as the average of 10000 repetitions of the experiment and expressed in micro-seconds.

## 4.2. CPU TIME VERSUS POLYHEDRA COMPLEXITY EXPERIMENTS

Tables 1 and 2 give the results corresponding to low and high complexity polyhedra respectively.

The CPU time that appears in the Tables is averaged from five different relative positions (by translation and rotation) of the polyhedra. In all the cases, we have made $a = b = c = 0.5$ in (10), but in one of the five experiments we have deformed one of the polyhedra making $c = 10$.

We compare QLDPA with the 'swap' algorithm (SA) (Llanas et al., 2000). This procedure was tested on the same examples as QLDPA.

The Tables use the following notation.

– N: Sum of the vertices of both polyhedra.

*Table 2.* High complexity experiments (time $\times 10^{-6}s$)

| N | $m$ | $n$ | $EPS$ | $t$ | PREC | TSA | NISA | TQLDPA | NDP |
|---|---|---|---|---|---|---|---|---|---|
| 1024 | 53 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 150 | 2 | 429 | 17 |
| 2024 | 103 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 164 | 2 | 435 | 17 |
| 4024 | 203 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 196 | 2 | 457 | 17 |
| 5024 | 253 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 207 | 2 | 466 | 17 |
| 6024 | 303 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 238 | 2 | 483 | 17 |
| 8024 | 403 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 261 | 2 | 529 | 17 |
| 9024 | 453 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 315 | 2 | 567 | 17 |
| 10024 | 503 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 342 | 2 | 613 | 17 |
| 11024 | 553 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 417 | 2 | 633 | 17 |
| 12024 | 603 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 434 | 2 | 683 | 17 |
| 14024 | 703 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 717 | 2 | 825 | 17 |
| 16024 | 803 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1000 | 2 | 1060 | 17 |
| 17024 | 853 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1069 | 2 | 1062 | 17 |
| 18024 | 903 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1238 | 2 | 1167 | 17 |
| 19024 | 953 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1326 | 2 | 1186 | 17 |
| 20024 | 1003 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1474 | 2 | 1288 | 17 |
| 21024 | 1053 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1603 | 2 | 1355 | 17 |
| 22024 | 1103 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1713 | 2 | 1437 | 17 |
| 24024 | 1203 | 10 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1974 | 2 | 1559 | 17 |

– $m, n$: Axial and radial subdivisions of the ellipsoid, for generating each $\mathcal{E}$-polyhedron.
– $EPS$: Stopping criterion.
– $t$: Parameter of SA and QLDPA.
– PREC: Precision obtained by SA and QLDPA. It depends on $EPS$ and $t$ and it is expressed as

$$|computed\ distance - exact\ distance|$$

– NISA: Number of iterations of the ‘swap‘ algorithm (every iteration consists of two projections).
– TSA: CPU time of SA.
– TQLDPA: CPU time of QLDPA.
– NDP: Number of distances between polygons (in $R^3$) evaluated by QLDPA.

In the first experiment we used tetrahedra, in the second parallelepipeds, in the third icosahedra and in the remainder $\mathcal{E}$-polyhedra with the values of $m$ and $n$ given in Table 1.

In all the experiments (Tables 1 and 2), the optimal value found for the parameter $t$ is 0. The number of iterations of SA increases with $t$.

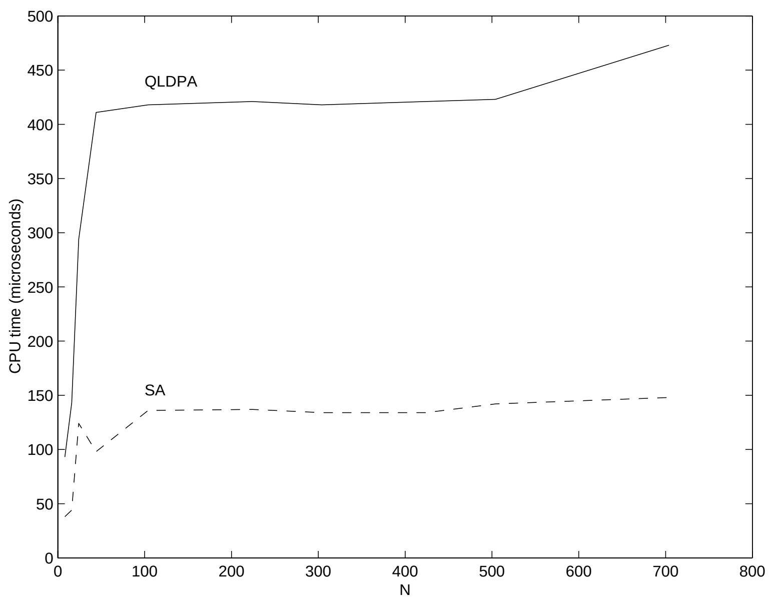

*Figure 7.* Low complexity.

The results of Table 1 can be shown as a graph of input complexity versus runtime. Here 'Input Complexity' is simply the sum of the number of vertices of the two polyhedra considered (N).

Figure 7 shows that, for low complexity problems, SA exhibits better time than QLDPA. This is due to the following

- In this range of complexity (0–700) more than 90 percent of the CPU time of QLDPA is used for computing the distance between polygons in $R^3$.
- The algorithm used for this task is the one proposed in Red (1983). Although exact, this method seems to be rather slow.

A comparison with the data appearing in other references is difficult because they have been obtained using different computers, operating systems, programming languages and representation of the polyhedra involved. We can estimate, with due precaution that, for low complexity problems, QLDPA is faster than algorithms based on classical optimization (Zeghloul et al., 1992) and Red's brute force method (Llanas et al., 2000). CPU time of QLDPA seems to be similar to the LC algorithm (Lin, 1993).

GJKC (Cameron, 1997) seems to be somewhat faster than QLDPA, but QLDPA could be accelerated using a faster algorithm than Red's for computing the distance between polygons in $R^3$.

The results of Table 2 have been obtained using pairs of $\mathcal{E}$-polyhedra with the same value of $m$ and $n$. They are shown in the following graphs, that represent the same data but using a different time scale.

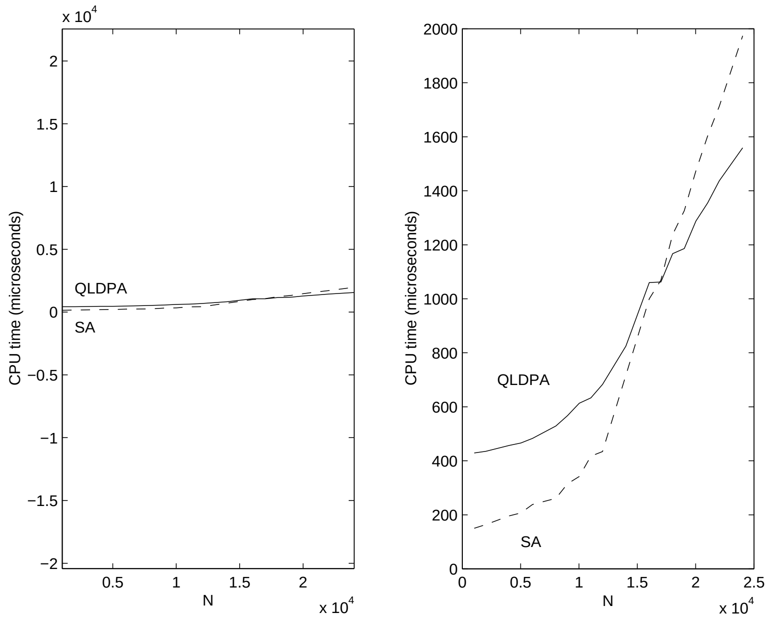

*Figure 8.* High complexity.

Figure 8 shows that QLDPA is faster than SA when the total number of vertices is greater than 17 000.

For very large polyhedra, the time that QLDPA needs for finding the initial pair of faces (one projection) is almost equal to the time used for computing the distance between polygons. SA needs to make at least four projections, so this increment of time originates the result shown in the graphs.

We have not found experimental data reported by other authors in this range of complexity.

In spite of the random character of the descent process, we have found in *all the experiments* made with the choice of the initial faces indicated in Section 4.1, that QLDPA *only performs the descent process*. It never performs a projection step (V).

## 4.3. CPU TIME VERSUS RELATIVE POSITION EXPERIMENTS

In this subsection we consider pairs of fixed polyhedra and study the variation of the CPU time of QLDPA and SA in relation to the relative position of the polyhedra. In the experiments, we have decreased the distance between the polyhedra keeping their orientation fixed.

The first experiment is shown in Figure 9.

The corresponding numerical results are given in Table 3.

The second experiment can be seen in Figure 10.

The corresponding numerical results are given in Table 4.

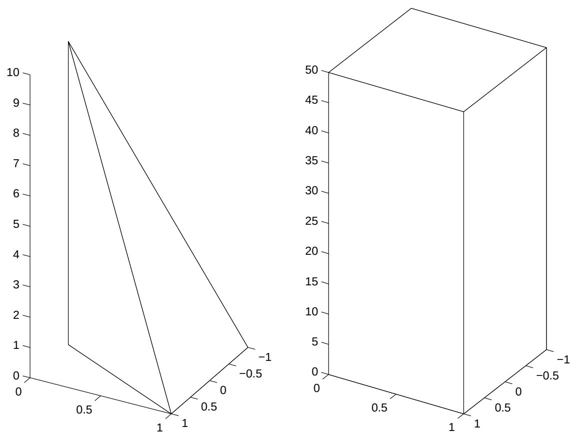

*Figure 9.* Polyhedra used in relative position experiments (1).

*Table 3.* Relative position Experiments (1) / (time $\times 10^{-6}s$)

| Distance | $EPS$ | $t$ | PREC | TSA | NISA | TQLDPA | NDP |
|---|---|---|---|---|---|---|---|
| 1 | $10^{-6}$ | 0 | $< 10^{-6}$ | 923 | 23 | 127 | 8 |
| 0.5 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1538 | 71 | 126 | 8 |
| 0.1 | $10^{-6}$ | 0 | $< 10^{-6}$ | 3878 | 182 | 126 | 8 |
| 0.01 | $10^{-6}$ | 0 | $< 10^{-6}$ | 8459 | 397 | 126 | 8 |
| 0.001 | $10^{-6}$ | 0 | $< 10^{-6}$ | 13243 | 626 | 126 | 8 |
| 0.0001 | $10^{-6}$ | 0 | $< 10^{-6}$ | 18257 | 857 | 126 | 8 |

The results of Tables 3 and 4 can be shown as graphs of distance versus runtime.

From Figure 11 it is clear that, when high precision is necessary, SA can present slow convergence. QLDPA always exhibits a fast behavior with a moderate dependence on the relative position of the polyhedra.

*Table 4.* Relative position experiments (2) / (time $\times 10^{-6}s$)

| Distance | $EPS$ | $t$ | PREC | TSA | NISA | TQLDPA | NDP |
|---|---|---|---|---|---|---|---|
| 1 | $10^{-6}$ | 0 | $< 10^{-6}$ | 703 | 23 | 434 | 18 |
| 0.5 | $10^{-6}$ | 0 | $< 10^{-6}$ | 1076 | 36 | 434 | 18 |
| 0.1 | $10^{-6}$ | 0 | $< 10^{-6}$ | 2384 | 81 | 439 | 18 |
| 0.01 | $10^{-6}$ | 0 | $< 10^{-6}$ | 4878 | 168 | 440 | 18 |
| 0.001 | $10^{-6}$ | 0 | $< 10^{-6}$ | 7552 | 261 | 434 | 18 |
| 0.0001 | $10^{-6}$ | 0 | $< 10^{-6}$ | 10194 | 353 | 439 | 18 |

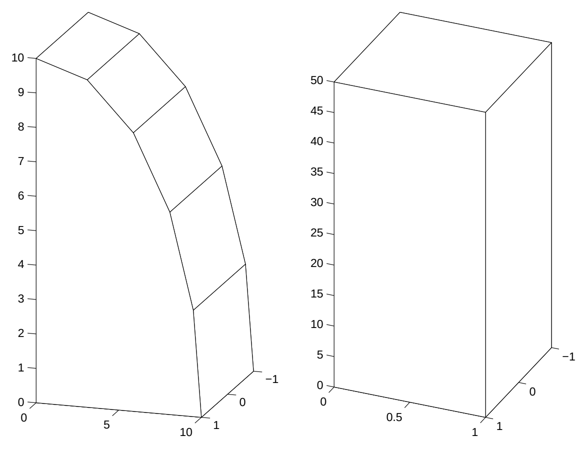

*Figure 10.* Polyhedra used in relative position experiments (2).

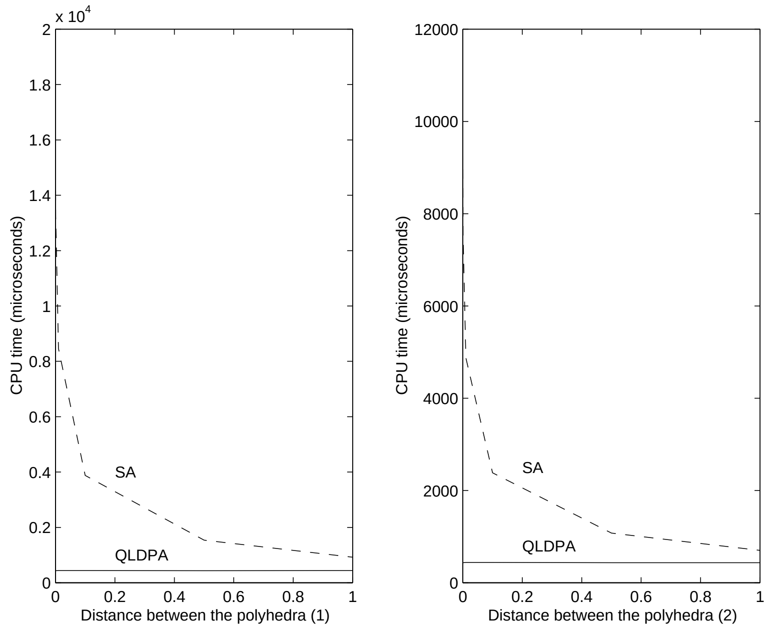

*Figure 11.* CPU time versus distance.

An important problem in applications is the tracking of the distance between polyhedra when they move. When the speed is low enough or the time interval is little enough, the tracking can be done by means of an incremental calculation. This means that the algorithm considers as initial features at the instant $t + \Delta t$ the nearest ones obtained at the instant $t$. We will call this procedure ‘with initialization’ (WI). When the speed is high or the time interval is large, we have to compute the distance without an initialization close to the solution. We denote this procedure ‘without initialization’ (WOI).

Although we have not studied the problem with initialization, QLDPA can be easily adapted to the incremental computation. In a dynamic environment the topological data are given at the beginning and they remain valid henceforth (if the polyhedra are rigid bodies).

The CPU time behavior of some fast algorithms is summarized in the following table.

| Algorithm | WOI | WI | $\Delta N$ |
|---|---|---|---|
| GJK | $O(N)$ | $O(N)$ | 8 – 500 |
| LC | $O(N)$ | $O(1)$ | 8 – 500 |
| GJKC | Very slow growth | $O(1)$ | 8 – 500 |
| QLDPA | Very slow growth | $O(1)$ | 8 – 24 000 |

By $\Delta N$ we denote the complexity range studied in every case.

## 5. Conclusion

This paper presents a sufficient criterion for the distance between the the faces of two convex polyhedra to be minimum. This result is applied to the design of a new algorithm (QLDPA) for computing the distance between convex and bounded polyhedra. The paper includes a wide set of numerical tests which show the performance of the algorithm and its advantages compared with SA and other algorithms.

- It has always obtained the exact solution, not an approximated one.
- When the complexity of polyhedra exceeds a given amount, QLDPA is faster than SA.
- The rate of growth of the CPU time of QLDPA is very slow.
- The dependence of QLDPA on the relative position of the polyhedra is rather moderate.

In dealing with low complexity problems QLDPA reduces the distance between polyhedra problem to the computation of a comparatively little number of distances between polygons in $R^3$. An optimal method for solving this last problem could give an optimal implementation of QLDPA. (An optimal method for computing the distance between segments in $R^3$ can be found in Lumelsky (1983)). Future research on this topic would be interesting.

## Acknowledgements

We thank the anonymous referees for the improvements they suggested. The research work presented in this paper was financially supported by the Consejería de Educación y Cultura de la Comunidad de Madrid and the electric company IBERDROLA.

## References

Balakrishnan, A.V. (1981), *Applied Functional Analysis*, Springer, New York, Heidelberg, Berlin.

Bobrow, J.E. (1989), A direct minimization approach for obtaining the distance between convex polyhedra, *The International Journal of Robotics Research* 8, 65–76.

Cameron, S. (1997), A comparison of two fast algorithms for computing the distance between convex polyhedra, *IEEE Transactions on Robotics and Automation* 13, 915–920

Dobkin, D.P. and Kirkpatrick, D.G. (1990), Determining the separation of preprocessed polyhedra-a unified approach, In: Paterson, M.S. (ed.) *Proc. 17th Internat. Colloq. Automata Lang. Program.*, Lecture Notes in Computer Science Vol 443, Springer, New York, Heidelberg, Berlin, pp. 400–413

Fujishige, S. and Zhan, P. (1990), A dual algorithm for finding the minimum-norm point in a polytope, *J. of the Operations Research Society of Japan* 33, 188–195.

Fujishige, S. and Zhan, P. (1992), A dual algorithm for finding a nearest pair of points in two polytopes, *J. of the Operations Research Society of Japan* 35, 353–365.

Gilbert, E., Johnson, D.W. and Keerthi, S.S. (1988), A fast procedure for computing the distance between complex objects in three-dimensional space, *I.E.E.E. Journal of Robotics and Automation* 4, 193–203.

Glaeser, G. (1994), *Fast Algorithms for 3D-Graphics*, Springer, New York, Heidelberg, Berlin.

Halperin, D., Kavraki, L. and Latombe, J.C (1997), Robotics, In: Goodman, J.E. and O'Rourke, J.(eds.), *Handbook of Discrete and Computational Geometry*, CRC Press, Boca Ratón–New York, pp. 755–778.

Hurteau, G. and Stewart, P. (1988), A distance calculation for imminent collision indication in a robot system simulation, *Robotica* 6, 47–51.

Khatib, O. (1986), Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research* 5, 90–98.

Lin, M. (1993), *Efficient Collision Detection for Animation and Robotics, Ph. D. Thesis,* University of California at Berkeley.

Llanas, B. and Moreno, C. (1996), Finding the projection on a polytope: An iterative method, *Computers Math. Applic.* 32, 33–39.

Llanas, B., Fernández de Sevilla, M. and Feliú, V. (2000), An iterative algorithm for finding a nearest pair of points in two convex subsets of $R^n$, *Computers Math. Applic.* 40, 971–983.

Lumelsky, V. J. (1985), On fast computation of distance between line segments, *Information Processing Letters* 21, 55–61.

Red, W. E. (1983), Minimum distances for robot task simulation, *Robotica* 1, 231–238.

Rockafellar, R. T. (1970), *Convex Analysis*, Princeton University Press, Princeton, NJ.

Sekitani, K. and Yamamoto, Y. (1993), A recursive algorithm for finding the minimum norm point in a polytope and a pair of closest points in two polytopes, *Mathematical Programming* 61, 233–249.

Vince, J. (1995), *Virtual Reality Systems*, Addison-Wesley, Reading MA.

Wolfe, P. (1976), Finding the nearest point in a polytope, *Mathematical Programming* 11, 128–149.

Zeghloul, S., Rambeaud, P. and Lallemand, J. P. (1992), A fast distance calculation between convex objects by optimization approach, *Proc. of the 1992 IEEE International Conference on Robotics and Automation,* (Nice, France-May 1992) pp. 2520–2525