Incremental Convex Minimization for Computing Collision Translations of Convex Polyhedra

Claudio Mirolo, Stefano Carpin, and Enrico Pagello

***Abstract*— The subject of this paper is an asymptotically fast and incremental algorithm for computing collision translations of convex polyhedra, where the problem at hand is reduced to determining collision translations of pairs of planar sections and minimizing a bivariate convex function. There are two main reasons, in our view, why the algorithm is worth consideration. On the one hand, the addressed proximity measure, namely collision translation, is not as widely studied as distance. On the other, its peculiar computation strategy may be interesting in itself, being well suited to work without initialization and also endowed with an inherently embedded mechanism to exploit spatial coherence. After outlining the main ideas of this novel approach and providing an estimation of the computational costs, we summarize a broad set of numerical experiments meant to explore extensively the behavior of the algorithm, both without and with initialization. Finally, in order to assess the efficacy and the potential of the approach under analysis, the attained performances are contrasted with those of other popular algorithms designed to compute distances between polyhedra. A thorough comparison of the reported query times and, more significantly, of the corresponding trends shows that the behavior of the collision translation algorithm is quite interesting, especially when used without initialization or under variable coherence, which should encourage further work on this approach.**

***Index Terms*— proximity problems, convex polyhedra, incremental algorithms, convex minimization, collision detection.**

I. Introduction

In this paper we discuss the structure and the performances of an asymptotically fast algorithm, with additional potential for incremental computations, designed to solve the following problem: *Given two convex polyhedra P, Q and a direction* **d**, *find the collision translation for P moving in direction* **d**. If P and Q do not collide, the algorithm returns suitable items proving the separation for all positions of P along its trajectory. Roughly speaking, the key idea characterizing our approach is that computing the collision translation of two convex bodies can be reduced to computing collision translations of pairs of planar sections and minimizing a bivariate convex function. This idea can be developed to design collision translation algorithms running in $O(\log^2 n)$ average time for a total number n of vertices, which corresponds to the best average-case complexity of the known techniques for answering similar proximity queries.

Analogously to the distance algorithms, the proposed algorithm could be appropriate as a basic operation to plan collision-free paths, both on-line and off-line, in the presence of fine-grain polyhedral descriptions of the objects. A complex representation of the workspace is indeed quite common for the geometric modelers based on CAD systems, as pointed out in [1], making it a critical goal to design fast techniques for computing proximity measures of primitive bodies. The algorithm may be especially useful to achieve more balanced performances across variable degrees of coherence [2], as it may be the case in real-time motion planning, but also the task of characterizing the configuration space off-line can be approached by systematically solving simple collision detection problems in order to probe the structure of the free space, for example via randomized sampling strategies.

One of the reasons of interest in this work, we believe, is that collision translation, unlike distance, has not received much attention in the literature, although it is straightforward to reduce to it other relevant proximity problems, such as intersection detection and collision depth in a given direction. A conceivable disadvantage is that the closest features are available only for the computed contact configurations. However, in some circumstances knowing the extent of a collision-free translation in the motion direction can be helpful to reduce the frequency at which the proximity measure must be sampled more than it would be possible with the distance. There are also specific tasks where a collision translation algorithm is more suitable. One such example can be found in the CAD programs that nowadays are applied to perfect parts design so that certain products can be easily assembled or disassembled. A typical case pertains to aircraft engines, because they need periodic inspections: in order to determine the feasibility of a quick maintenance plan, it is critical to know the clearance in certain directions rather than in absolute terms.

A first step along the line of research investigating the power of our novel approach was the algorithm analyzed in [3], in regard to which the present work introduces a few major technical improvements as well as provides a richer collection of data as a basis for the assessment. More specifically, the main original contributions of this paper are:

(i) the characterization of the faceted structure of the convex function's graph and its application to sharpen the minimization process;
(ii) the introduction of a mechanism for exploiting coherence in order to speed up the computation;
(iii) a broad experimental analysis, including a comparison with other important proximity algorithms.

The deeper connection of the minimization process with the shape of the convex function, the tools developed to solve the

Claudio Mirolo is with the Department of Mathematics and Computer Science of the University of Udine, Italy. Stefano Carpin is with the School of Engineering of the University of California, Merced, USA. Enrico Pagello is with the Department of Information Engineering of the University of Padova, Italy.

related subproblems and the significant refinement (ii) make the new algorithm different in several important respects from that outlined in [3].

The $O(\log^2 n)$ complexity bound refers to computations *from scratch*, i.e., without initialization, meaning that no previous proximity information is either available or exploited. On the other hand, it is typical of the applications in the fields of on-line motion planning, simulation, animation and computer aided design, that a huge number of proximity tests have to be carried out after subsequent short movements of the objects in the workspace. In such situations the performances of the procedures answering basic geometrical queries are crucial, and exploiting the *spatial* and *temporal coherence* may be beneficial in order to speed up the computations, since the outcomes of the proximity tests on close workspace states are likely to be represented by close features. Efficient algorithms designed to process the knowledge of a previous result (initialization) are referred to as *incremental* algorithms.

Besides addressing a less customary proximity measure than distance, the core approach of the proposed algorithm is quite peculiar, suitable both to work without initialization (computations *from scratch*) and to exploit spatial coherence (*incremental* computations) with the aid of a mechanism that is inherently embedded in the computation strategy. The very nature of the approach allows us to endow the algorithm with a self-tuning capability, at negligible additional costs even for unrelated collision tests, by simply refining the choice of the splitting points during the minimization process. The behavior of the algorithm is substantiated by a set of numerical experiments, whose results can be summarized as follows:

- The performance trend is good when collision translations are computed from scratch and the algorithm tends to be even faster if the input polyhedra do not collide.
- The average number of minimization steps decreases regularly, as expected, and gets very close to the smallest possible value, while the coherence is growing higher.
- The behavior without initialization is quite interesting also if compared with that of other popular proximity algorithms, whereas the incremental performances could probably be improved.

The performances have been compared with those of two well known algorithms designed to compute distances between polyhedra: Cameron's enhanced GJK [4] and Larsen, Gottschalk, Lin and Manocha's PQP [5]. Moreover, the capability to adapt to variable coherence has been investigated in light of Guibas, Hsu and Zhang's results on *H-Walk* [2], an algorithm of specific interest for this feature, by testing our algorithm under conditions corresponding to the experiments discussed in [2]. Although these "yardsticks" solve a different proximity problem and although [5] applies to more general polyhedra, the comparative results contribute to a first assessment of the effectiveness and potential of the approach under analysis. All the algorithms, indeed, share the same kind of application frameworks, and answering distance or collision translation queries appears to require a similar computational effort, as also suggested by the known asymptotic bounds.

Related work

Starting from the early work on the geometric models of robot workspaces, there has been a steady interest in proximity measures and properties [6], since the design of efficient algorithms to answer these kinds of queries is generally thought to be critical to the development of effective tools for motion planning, as pointed out in [7], and for a variety of applications in other fields, such as simulation, animation, virtual reality [8], where geometry plays an important role. Several algorithms proposed in the literature apply to convex models, a typical choice as basic objects components, e.g. [9], [10], [11], [12], [13]. In the usual case of convex polytopes, the best asymptotic bound for various proximity problems (intersection detection, collision detection, distance, depth of collision) is the $O(\log^2 n)$ worst-case bound attained by exploiting the hierarchical representations of polyhedra with $O(n)$ vertices, [14], [15]. Some efforts, however, have been addressed to the case of more general bodies: to define different measures of penetration [16]; to speed up the interference tests for elementary polyhedral items (vertices, edge, triangular faces) [17] or for other bounding surfaces [18]; to deal with certain kinds of movement [19], [20].

Since the work by Lin and Canny in the early '90s [21], the complexity of the settings arising in the application fields mentioned above has also fostered research on incremental algorithms being able to exploit the coherence and run in nearly constant time per query, e.g. [4], [22], [23], [24], [25], [26], [27], or even to adapt to variable coherence [2]. A related issue is the design of suitable representations to speed up the broad phase aimed at selecting few pairs of primitive volumes (e.g., convex polyhedra) for the proximity tests. Examples of this approach are the application of kinetic data structures [28], as well as of other hierarchies based on bounding volumes of simple shape, which are considered in several papers, e.g. [8], [29], [30], [31], [32], [24], [33], [34], [35], [36].

As already said, [4], [5] and [2] are of specific relevance for the analysis presented in this paper. Besides having analogous purposes, the reasons of our interest in the [4] are manifold: its scope is somehow homogeneous with that of the collision translation algorithm; it runs very fast in practice and can exploit the space coherence; further comparisons with related techniques are already available, e.g. [4], [26], [37]. Similarly, [5] is also widely known and is another natural candidate as a benchmark in the field of proximity algorithms, as can be seen e.g. in [38]. Finally, [2] is a key yardstick for its flexibility under different degrees of coherence.

Organization of the paper

The paper is organized as follows. In section II we present the key ideas characterizing the approach to computing collision translations. Then, in section III we outline the structure of the algorithm and show how the spatial coherence can be exploited. The "yardstick" algorithms for the performance comparison are the subject of section IV, where we also mention the reasons of this choice. Finally, in section V the main experimental results are summarized and analyzed.

II. Grounds of the approach

Our approach builds upon the work done in [3], but with a few important extensions. The principal refinements include: the local characterization of the polyhedral graph of the convex function; the adoption of a new polygonal cell-subdivision with cell-shift operations, instead of an isothetic grid, as a discrete minimization structure; the incremental techniques aimed at controlling the focus of the minimization relative to a previous solution. Moreover, the combinatorially complex algorithms dealing with polygon-drum and drum-drum pairs, while essential with an isothetic grid, can now be replaced by simpler tools. Thus, after briefly summarizing, in the next subsection, the key results from [3], we will then proceed by analyzing some useful relationships between the configurations of pairs of planar sections and the linearity properties of the convex function. Indeed, a deeper understanding of such relationships is at the root of the major improvements of the minimization process.

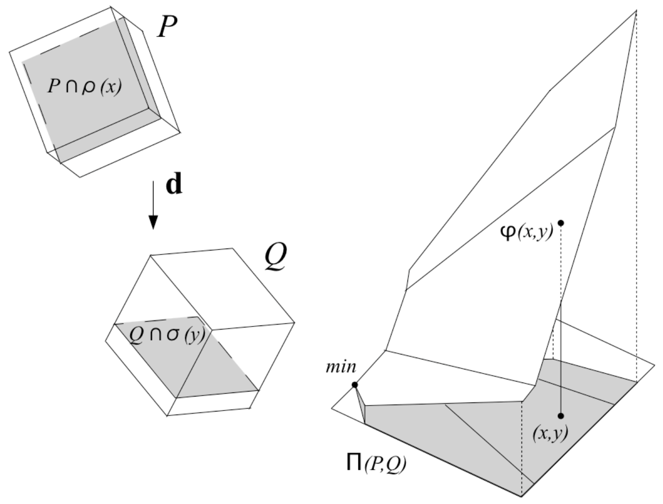

Fig. 1. Illustration of the relationship between collision translations and convex minimization for a pair of cubes. The graph of φ is represented by the white faceted surface on the right; $Dom(\varphi)$ is gray-shaded.

A. Collision translations and convex minimization

To begin with, we recall the most basic result: collision translations for two convex bodies P and Q can be reduced to minimization of a bivariate convex function that represents collision translations of pairs of planar sections of P and Q. More formally, we can prove the following proposition [3]:

> Let P and Q be two closed and bounded convex regions, **d** a direction in the space, $\{\rho(x)|x \in \mathbb{R}\}$ and $\{\sigma(x)|x \in \mathbb{R}\}$ two independent families of parallel planes. Then
>
> $$\varphi(x, y) = coll_d(P \cap \rho(x), Q \cap \sigma(y))$$
>
> is a convex function with bounded domain in $\mathbb{R}^2$.

In the above statement, $coll_d(X, Y)$ denotes the extent of the collision translation in direction **d** for X and Y, i.e., the least $\tau \in \mathbb{R}$ such that $dist(\{p + \tau\mathbf{d} \mid p \in X\}, Y) = 0$; $\rho(x)$ and $\sigma(y)$ identify the planes by their distances $x, y \in \mathbb{R}$ from two independent reference planes. Based on the definition of $coll_d$, also negative values make sense, and $coll_d$ is undefined only if there do not exist (positive or negative) displacements of X in direction **d** such that X and Y intersect. Throughout the paper we will often refer to collisions of bodies, sections, or related items: unless otherwise specified, in connection with the terms *collision* and *collide* we always imply "*by translation* in direction **d**," in the sense of $coll_d$'s definition.

Clearly, if a collision translation for P and Q is defined, then its extent is the minimum of φ:

$$coll_d(P, Q) = \min\{\varphi(x, y) \mid (x, y) \in Dom(\varphi)\}$$

and the planar sections corresponding to the minimum contain the contact points. The meaning of the proposition is illustrated in figure 1. It tells us that if we are able to compute collision translations for pairs of polygons in the space, then we can determine the collision configuration of two convex polyhedra by standard convex minimization techniques such as the method of centers of gravity [39]. Basically, these techniques work by repeatedly splitting and cutting off a slice of a convex region containing the point of minimum. At each step the cut line is constrained to pass through the centroid of the region, to assure a balanced bisection, and it is in fact a support line for the corresponding level curve of the convex function.

This is not yet satisfactory, since it does not guarantee that we can find an exact solution in a finite number of steps. Thus the next question is how to transform a search problem on a continuous domain into a search problem on a discrete domain. Following [3], such a discrete domain can be defined as the set of rectangles of an isothetic grid. Then, under reasonable assumptions, the grid rectangle containing the point of minimum can be found in $O(\log k)$ minimization steps in the average, where k is the size of the grid. After solving a few related problems and putting all the pieces together, we end up with an algorithm that computes collision translations for pairs of polyhedra with $O(n)$ vertices in $O(\log^2 n)$ time in the average and $O(\log^3 n)$ in the worst case.

B. Discrete structure of the convex function

A deeper analysis of the properties of the convex function φ allow us to take a further step forward. Since P and Q are polyhedra, φ's graph is also faceted and its topology projects into a corresponding polygonal partition of $Dom(\varphi)$. Such a partition is more appropriate than the isothetic grid [3] as a discrete structure to search for the point of minimum. For our purposes, however, it is necessary to extend the partition outside $Dom(\varphi)$, to cover the whole rectangular region

$$\Pi(P, Q) = \{(x, y) \mid (P \cap \rho(x) \neq \emptyset) \wedge (Q \cap \sigma(y) \neq \emptyset)\}$$

representing all possible pairs of planar sections of P and Q.

It is possible to achieve this goal by introducing suitable invariants characterizing the orientation of the cut lines built in the minimization process. Such straight lines are either perpendicular to φ's gradient, if drawn through points within the domain, or do not intersect $Dom(\varphi)$, if the cut points fall outside the domain, and in both cases we want their orientation

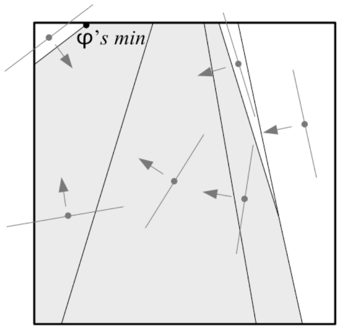

Fig. 2. Mark of the convex function of figure 1. The invariant orientation of the cut lines is drawn at a sample point in each cell; the arrows represent the gradient vectors within $Dom(\varphi)$ (gray cells) or the normal directions towards $Dom(\varphi)$ in the outer cells.

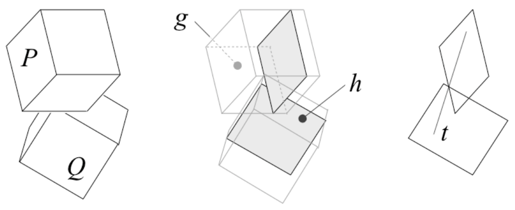

Fig. 3. Illustration of an *edge contact* instance for the cubes P and Q in the configuration of figure 1. The cubes and their planar sections are drawn as they appear by looking in direction **d**; the straight line t is the intersection between the section planes in the contact configuration.

be the same for all points in a *cell*, i.e. a region of the partition. We will refer to the polygonal cell decomposition of $\Pi(P, Q)$ generated by these invariants as the *mark* of φ. For instance, figure 2 shows the mark of the convex function visualized in figure 1 and the orientation of the cut lines within each cell.

The local properties of the convex function φ in a neighborhood of the point (x, y) can be determined from the output of the algorithm used to compute collision translations of pairs of planar sections (see section III). Given the information on either the contact or the separation of $P \cap \rho(x)$ and $Q \cap \sigma(y)$, we can build a cut line through (x, y) and, in addition, we can find a more favorable point in the corresponding cell of the mark, i.e., a point such that a cut line with the same orientation gets closer to the point of minimum. It is also important to notice that if we are able to find the *most* favorable point in a cell, then the whole cell can be discarded from further consideration (see also figure 9).

Three types of cells characterize the mark, according to the possible outcomes of the algorithm for computing collision translations of pairs of planar sections:

a) *Edge contact* – the planar sections $P \cap \rho(x)$ and $Q \cap \sigma(y)$ collide by translation in direction **d** and their contact items are both edges, as illustrated by the drawings in figure 3. Since such edges are cords of two faces, say g of P and h of Q, by linearity all the pairs (u, v) such that the contact items of $P \cap \rho(u)$ and $Q \cap \sigma(v)$ are $g \cap \rho(u)$ and $h \cap \sigma(v)$, respectively, lie in the projection of the same facet of φ's graph. All the gray cells in figure 6 are related to contacts of this kind, where the pair of faces (g and h) changes from cell to cell.

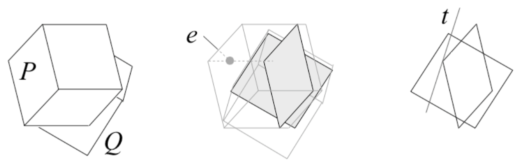

Fig. 4. A *vertex contact* instance for a configuration of the cubes where P in is translated with respect to the scene of figure 3. Again, **d** is the view direction and t is the intersection of the section planes in the contact configuration.

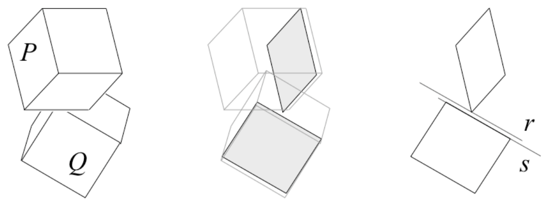

Fig. 5. A *separation* instance for the situation of figure 1, viewed in direction **d**. In this case it is convenient to think of the straight lines r and s as the intersections between $\sigma(y)$ and the separating planes (parallel to **d**).

b) *Vertex contact* – a vertex of the planar section $P \cap \rho(x)$ hits the plane $\sigma(y)$ inside Q by translation in direction **d**, as illustrated in figure 4, or symmetrically. Let e be the edge of P containing the contact vertex, by linearity all the pairs (u, v) such that $P \cap \rho(u)$ collides with the interior of $Q \cap \sigma(v)$ at the vertex $e \cap \rho(u)$ fall in the projection of the same facet of φ's graph.

c) *Separation* – the planar sections $P \cap \rho(x)$ and $Q \cap \sigma(y)$ do not collide by translation in direction **d**, as in the situation depicted in figure 5, thence are separated by a pair of planes through their vertices, which are parallel to each other and also to **d**. In the following, we will refer to such an arrangement of planes and polygons as a *separation construction*. In these cases a cell is defined in such a way that it contains all pairs (u, v) corresponding to planar sections whose separation is witnessed by couples of planes with the same orientation and through vertices on the same two edges of P and Q. In particular, these invariants apply to the white cells in figure 6.

A qualitative characterization of the mark cells is shown in figure 6; the drawings should be self-explanatory based on the comments to the figures 3 and 5.

III. Outline of the Algorithm

Figure 7 outlines the general structure of the algorithm designed to compute collision translations for two polyhedra P, Q and a translation direction **d**. Its core is the minimization loop based on the properties discussed so far, which is more suitable to implement and behaves better than the solution proposed in [3]. In this respect, without initialization, the average number of minimization steps reduces by 15-20%, whereas the number of invocations of $O(\log n)$ algorithms, a more precise performance measure, decreases by 9.6-10.2 calls almost independently of the polyhedron complexity. We begin by considering computations without initialization and postpone the incremental case until section III-D.

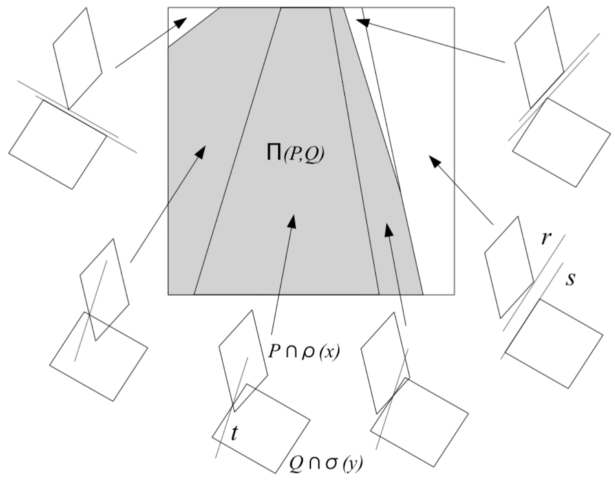

Fig. 6. Qualitative characterization of the mark. The sample sections of the cubes P and Q are drawn as they appear by looking in direction **d** for the setting of figure 1. The cells in $Dom(\varphi)$ are related to *edge contact* instances; those outside the domain refer to *separation* instances.

```
input :
    two convex polyhedra $P$,$Q$ and a direction d ;
    if available, the previous solution $p$ and change $\delta$ ;

1   $M$ := rectangle of all pairs of planar sections of $P$, $Q$ ;
2   if  $p$ is provided   then  $c := p$
3                         else  $c :=$ centroid of $M$ ;
    loop
4       $s :=$ cut line through $c$ ;
5       cell-shift $s$ to point $q$ ;
6       if  $q$ solves the problem   then   exit ;
7       update $M$ w.r.t. $s$ ;
8       $c :=$ centroid of $M$ ;
9       if appropriate, update $c$ w.r.t. $N^{\delta}(p)$
    end ;

output :
    collision translation $\varphi(q) = coll_d(P,Q)$
```

Fig. 7. Structure of the algorithm for computing collision translations. When working without initialization, the assignment in line (2) and the statement in line (9) are never executed. For simplicity, the output refers only to the situations where a collision translation is defined.

A. Computations from scratch

Refer to figure 7. At line (1) the region M is created to represent the rectangular set of all pairs of planar sections and, when working without initialization, (3) the centroid c of M is computed. Then, at each iteration of the minimization loop: (4) the cut line s through c is computed and (5) shifted to a better point q in the same cell of the mark; if q is the solution (6) the algorithm ends; otherwise (7) the region M is updated by cutting off a slice through s and (8) the centroid is recomputed. Eventually, q is recognized to be either the point of minimum or a witness proving that φ's domain is empty. In both cases the solution is found.

To estimate the computational costs, we can reasonably envisage that the average number of minimization steps grows as the logarithm of the total number of cells C, i.e., $k = O(\log C)$, since the number of cells which overlap with the region M tends to be proportional to the area of M and about half of the area of M is discarded at each step. For technical reasons, it is convenient to provide a slightly *finer* characterization of the mark cells, in order to be able to locate in $O(\log n)$ the best point where to shift the cut line at each minimization step (line 5 of the pseudocode). More specifically, we can think of P and Q as sliced into drums, i.e. polyhedra whose vertices lie on two parallel planes [40], by the section planes through the vertices. Then we consider the cell decomposition resulting from all the trapezoidal faces of the drums, as well as the corresponding edges and vertices, instead of the original ones. Polyhedra with $O(n)$ vertices can be sliced into $O(n)$ drums bounded by $O(n^2)$ trapezoidal faces. Since a cell is related to one or two items among faces, edges, or vertices, as seen in section II-B, their overall number is $C = O(n^4)$ and we have $k = O(\log n^4) = O(\log n)$.

Now consider the generic iteration step in the schema of figure 7. The cost of line (4) is $O(\log n)$ and comes from the computation of collision/separation configurations of planar sections that will be introduced in a moment. Also the computation of point q in line (5), to be outlined in section III-C, may require $O(\log n)$, namely for the *vertex contact* case, the other two cases being processed in constant time. After $j \leq k$ minimization steps, the region M is bounded by at most j sides, thus the cost of lines (7) and (8) is $O(k)$ in the worst case. Finally, the actions in line (6) and (9) can be done in constant time, the latter being only relevant for incremental computations. This allows us to conclude that the algorithm runs in $O(log^2 n)$ time in the average.

B. Collision translations of planar sections

Following the approach outlined above, a basic subproblem to be solved concerns pairs of planar sections:

> Given two convex polygons $R = P \cap \rho(x)$ and $S = Q \cap \sigma(y)$ and a direction **d** in the space, compute either the *collision configuration* or a *separation construction* for R moving in direction **d**.

What is most important here, is that this task can be accomplished in $O(\log n)$ in the worst case for two polygons with $O(n)$ vertices. For a detailed description of the algorithm we refer to the technical report [41]. However, the reader may gain some insights by figuring how the spatial problem can be transformed into an equivalent planar problem.

To this aim, let us assume for convenience that the reference planes $\rho(0)$ and $\sigma(0)$, and the direction **d** are in *general configuration*, which means that no pair of these items are parallel to each other. Under this assumption the planes $\rho(x)$ and $\sigma(y)$ intersect along a straight line t and R sweeps a polygon R' on $\sigma(y)$ while moving in direction **d**, as shown by the construction in figure 8. Said otherwise, R' is the projection of R along **d** onto the plane $\sigma(y)$. Clearly, R and S collide if and only if $R' \cap S \neq \emptyset$. Moreover, while R moves in direction **d**, the straight line t shifts towards S in the plane $\sigma(y)$ and the contact configuration corresponds to the situation where t reaches a first vertex of $R' \cap S$. If, on

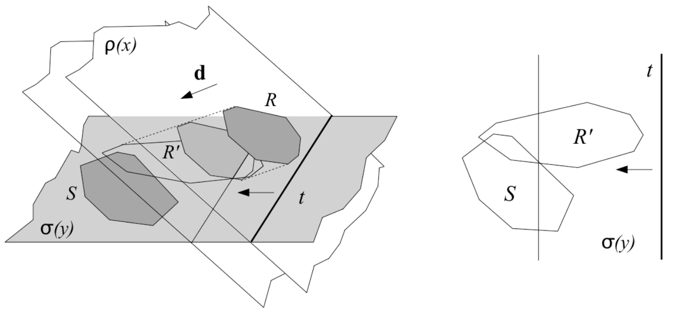

Fig. 8. Collision translations of pairs of planar sections can be reduced to equivalent two-dimensional problems by considering the region R' swept by R on S's plane $\sigma(y)$. The arrangement in this plane is drafted on the right.

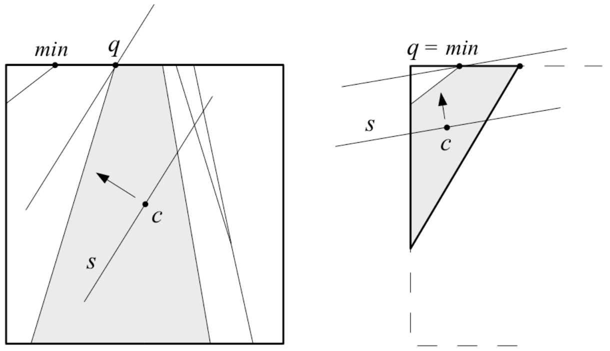

Fig. 9. Two minimization steps for the configuration of figure 1. At each step: first the cut line s is computed at the centroid c of the search region; then a more favorable cut point q is determined and the cut line is shifted there; finally, the search region is split. In this example the solution is found during the second step.

the other hand, $R' \cap S$ is empty and the polygons do not collide, we know that $(x, y) \notin Dom(\varphi)$. In this case the two-dimensional algorithm returns two parallel support lines separating R' and S, like r and s in the figures 5-6, i.e. two parallel straight lines through vertices of R' and S such that the polygons lie in disjoint halfplanes. Thus, the planes parallel to **d** containing the support lines define a separation construction for the polygons R and S, which is the suitable information to be exploited to locate $Dom(\varphi)$ with respect to (x, y).

The planar problem can be solved by binary search: the sides of the polygons R' and S are searched for the point where t hits $R' \cap S$, which may be either the intersection of two sides (*edge contact*) or a vertex of one polygon lying inside the other (*vertex contact*), possibly ending with a couple of separating support lines instead (*separation*). It should be noticed that the logarithmic cost of solving the planar problem (for R' and S) also applies to the original spatial problem (for R and S) provided the representation of R' is not completely built in advance, but only those projected items that need to be processed are actually computed, and in constant time, from the corresponding items of R.

C. Cut lines and cell-shift of a cut line

Starting from the information about either the contact or the separation of two planar sections $P \cap \rho(x)$ and $Q \cap \sigma(y)$, it is possible to compute in constant time the orientation of the cut line at point (x, y) for the next minimization step. We refer again to [41] for the geometric constructions useful to understand how the configurations of pairs of polyhedra relate to cut lines, but we consider instead the problem of determining a convenient cut point. More precisely, our purpose is now to compute a point $q = (x', y')$ where to shift the splitting line in order to discard the whole cell from the updated search region M, as shown in figure 9. As mentioned before, in order to solve any instance of this problem in either constant or logarithmic time, we define a *finer cell* by imposing further constraints (trapezoidal subfaces). As a consequence, each *intrinsic* cell may contain a few finer cells, but in the essence we save all the intended benefits of the mark.

For a sketch of how (x', y') can be computed, we refer again to the three situations considered in section II-B:

a) *Edge contact*. For all (u, v) in a cell, the contact points of $P \cap \rho(u)$ and $Q \cap \sigma(v)$ belong to two faces g and h, hence the least $\varphi(u, v)$ is the collision translation of g and h themselves when g moves in direction **d**. From the contact points of g and h we can easily determine the section planes $\rho(x')$ and $\sigma(y')$ and the point (x', y'), which must lie on the cell boundary. Since the finer cells are defined for trapezoidal *drum* faces, the collision configuration of two such faces can be computed in constant time.

b) *Vertex contact*. For all (u, v) in a cell, the contact points are a vertex of $P \cap \rho(u)$ on P's edge e and a point inside $Q \cap \sigma(v)$, or a symmetric configuration. In this case the finer cell is defined relative to a maximal drum D of Q, and the least value of $\varphi(u, v)$ corresponds to the collision translation of e and D, for e moving in direction **d**, which can be computed in logarithmic time on the number of drum faces via binary search. From the contact between the edge and the drum we can determine two section planes $\rho(x')$ and $\sigma(y')$ and the point (x', y') on the boundary of the finer cell.

c) *Separation*. For all (u, v) in a cell, the separation is witnessed by two vertices of $P \cap \rho(u)$ and $Q \cap \sigma(v)$, belonging to P's edge a and Q's edge b, and by two corresponding straight lines $r(u)$ and $s(v)$ with a given fixed orientation on the plane $\sigma(v)$. This guarantees that we can build cut lines with a fixed orientation as well. This time (u, v) reaches the boundary of the finer cell when either $r(u)$ and $s(v)$ overlap or one of the contact points is an endpoint of a or b. All the information is then provided by two edges and the orientation of the separating lines, which allows us to compute (x', y') in constant time.

D. Exploiting spatial coherence

In a variety of applications, including on-line motion planning, a proximity measure needs to be recomputed after small intervals of time. Therefore, also the movements of the objects between two subsequent time steps should not be too large and we can expect that their relative configurations do not change much. This observation applies, in particular, to the closest points realizing the minimum distance, as well as to the contact points in the collision configurations, after subsequent proximity tests. In similar situations we can gain

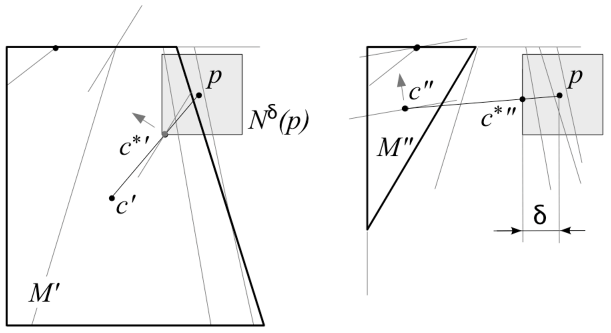

Fig. 10. Minimization steps with initialization p for the configuration of figure 1: the focus neighborhood is $N^\delta(p)$. After the first step, starting at p, the search region is M' with centroid c', the intersection $c^{*\prime}$ between $c'p$ and the boundary of $N^\delta(p)$ falls inside M'; so, the next cut point is $c^{*\prime}$. Then, the updated search region is M'' with centroid c'', $c''p$ intersects $N^\delta(p)$ outside M'', and the minimization proceeds at c''.

considerable speed-up by exploiting the information on a previous computation of the same proximity measure. As said before, algorithms designed to this purpose are referred to as *incremental* algorithms in the literature.

The very nature of the approach outlined above makes it possible to endow the collision translation algorithm with a flexible mechanism to exploit spatial coherence, which rests on a simple idea: during the minimization process, we can try to focus the search for the point of minimum in a suitable neighborhood of a previous solution. In order to implement this idea, we have to address two problems: (i) how to choose a suitable neighborhood and (ii) how to recover if the solution lies outside of it. As far as problem (i) is considered, our *focus neighborhood* is simply an isothetic square $N^\delta(p)$ of size 2δ, centered at the previous solution p, where δ is heuristically related to the changes of the test configuration, i.e., positions and orientations of P, Q and **d**. A suitable choice for δ is the extent of the component perpendicular to **d** of the shift of P's contact point. (Notice that we are not assuming that P is *actually* moving in direction **d**.)

However, there is no guarantee that the next solution will fall in the chosen neighborhood. So a mechanism for switching to the standard search strategy must be provided, in which case it would also be desirable to save the work already done. The implemented technique addresses problem (ii) as follows. The search starts at the point p representing the previous solution, but the minimization region M is initialized as usual (line 1 in figure 7). At each step, if the centroid c of the minimization region does not fall inside the neighborhood $N^\delta(p)$ centered at p, we consider the intersection point c^* between the boundary of $N^\delta(p)$ and the straight line segment cp. If c^* lies in M then it is chosen as the next cut point; otherwise the next cut point is the centroid c and we forget the neighborhood. Figure 10 illustrates two subsequent minimization steps with focus neighborhood $N^\delta(p)$; the latter step resumes the standard process since $c''p$ does not intersect the boundary of $N^\delta(p)$ within the search region M''.

The incremental behavior of the algorithm is achieved simply by the operations in lines (2) and (9) of figure 7: initially (2) the previous solution is chosen as first cut point and at the end of each iteration (9) the cut point may be c^* instead of the centroid c of M. It is worth observing that the search focus can be tuned by means of the parameter δ: the closer two consecutive configurations are, the faster the proximity measure can be updated.

A rough estimate of the computational costs can be obtained as follows. For simplicity, suppose that M is initially a square and call γ the ratio between δ, measuring the configuration change, and M's side length. The expected number of cells intersecting the neighborhood $N^\delta(p)$ over the total number C of cells is about

$$Area(N^\delta(p)) \;/\; Area(M) \;=\; 4\gamma^2$$

and then the number of minimization steps for a search bounded within $N^\delta(p)$ should be proportional to

$$\log C \;-\; 2\log(1/\gamma)$$

So, if the updated solution lies inside $N^\delta(p)$, the gain with respect to the standard strategy is of about $\Theta(\log(1/\gamma))$ iterations in the average. Since this rough estimate appears to be in good accordance with the experimental trends, this means that the updated solution falls inside $N^\delta(p)$ with high probability and witnesses the important role that the space coherence may play.

IV. "Yardstick" algorithms

We now introduce the algorithms that we have considered in order to try a first appraisal of the performances attainable with the approach described in the previous section: the extended GJK, the distance computation procedure available in the Proximity Query Package, and the Hierarchical Walk. As said before, the comparisons are not completely fair since these algorithms answer different proximity queries, i.e. distances rather than collision translations. Moreover, the broader applicability of PQP, not restricted to convex bodies, should be taken into account, but in this case we are also interested in achieving a better understanding of the power of hierarchical structures as opposed to convexity properties, when the latter could be exploited as well. In short, since the bulk of the experiments discussed in the literature are relative to distance computation, the results of such experiments also provide the natural benchmarks against which to compare new results. Although the tools are not fully equivalent, we think that this kind of comparisons make sense and can hopefully suggest possible directions of future work.

A. Enhanced GJK: convex polyhedra and spatial coherence

The first yardstick is the enhancement proposed by Cameron [4] of the classical Gilbert, Johnson and Keerthi's (GJK) algorithm [9]. The original GJK technique computes the distance between two convex polyhedra by finding the distance from the origin of their Minkowski difference. To this aim, a simplex is maintained and iteratively checked for optimality, i.e. to see whether it minimizes the distance from the origin, and possibly updated. The strength of the original algorithm lies in its efficiency to determine if a simplex is optimal, and, if not, to find a better one.

As shown in [4], this approach can be improved by applying a hill climbing technique while looking for a better simplex. The key observation is that the hill climbing step can be sped up by providing a suitable starting point, called a *seed*. Moreover, the data produced while checking for optimality can act as good seeds. In practice, the enhanced GJK algorithm returns the distance in nearly constant time when small changes in the relative configuration arise. In [4] it is also argued that under these hypotheses the behavior of the enhanced algorithm tends to be similar to that of the incremental technique [21].

There are two main reasons of our interest in considering the enhanced GJK algorithm. On the one hand, it runs very fast in practice, and under similar conditions with respect to our algorithm: the input polyhedra are required to be convex and it is well suited to exploit coherence. On the other hand, plenty of experimental data are available, which compare variants of the GJK scheme, including the enhanced GJK, with other approaches to distance computation, e.g. [4], [37], [26].

B. PQP: general polyhedra and hierarchical structures

The second yardstick belongs to the PQP software library [5], including algorithms to answer three types of queries: interpenetration detection, approximated separation distance and exact separation distance. Like other prior techniques, PQP exploits a bounding volume hierarchy to speed up the computation. The idea underlying bounding volumes is quite simple: each object is bounded by a certain shape for which the query at hand can be easily answered. Such containers are organized into hierarchical structures, usually trees, and the volumes found at deeper levels are smaller and approximate better the actual shape of (parts of) the object; eventually, the leaves represent exactly the components of the objects, and then allow to provide as accurate an answer as needed.

Different kinds of bounding volumes have been considered in the literature, among which we can mention axis aligned bounding boxes, oriented bounding boxes and bounding spheres. Some representations are also based on hybrid bounding hierarchies, i.e., different shapes are used at different levels of the tree. PQP exploits oriented bounding boxes for interference detection and swept spheres for distance computation. The latter, in particular, come in three forms: point swept spheres, line swept spheres, and rectangle swept spheres.

In order to understand correctly the results discussed in this paper, it is important to recall that PQP's input models are described in the very general form referred to as *triangle soup*. After all the input triangles of a model have been provided, PQP builds the suitable hierarchies in a preprocessing phase, and then becomes ready to answer multiple queries. It should also be noticed that PQP does not exploit coherence. Nevertheless, PQP is widely known and is a natural candidate benchmark in the field of proximity algorithms. Also for PQP experimental comparisons with related tools are available, e.g. in [38] it is argued that in most situations PQP's performances match those of the fastest algorithms. One further reason for considering PQP lies on its use of hierarchical structures and bounding volumes, which makes a performance comparison with this approach interesting in itself.

C. H-Walk: convex polyhedra and variable coherence

Guibas, Hsu and Zhang designed *H-Walk* [2], that combines the advantages of the algorithms proposed by Dobkin and Kirkpatrick [14] and by Lin and Canny [21]. Specifically, Lin and Canny introduced the key idea of exploiting the *spatial coherence*, by observing that, when the polyhedra are moving and a query is asked frequently, the two closest features (points, edges or faces) can easily be updated starting from the former pair of closest features. Their algorithm starts from the previous closest features and *walks* (traverses) different pairs of features until the new solution is found. Since the length of the walk is also an accurate measure of the computational costs, it follows that the time requirements are almost constant for high levels of coherence, but deteriorate seriously (up to quadratic complexity in the number of vertices) if there are jumps between subsequent configurations.

On the other hand, Dobkin and Kirkpatrick introduced a preprocessed representation of the convex polyhedra that allows to answer a variety of proximity queries in polylogarithmic time. The preprocessing step can be carried out in linear time and builds a layered hierarchy approximating the solid body from the interior. The original algorithm, however, does not capitalize on the information gathered from formerly solved instances of the proximity problem, every computation being performed from scratch. The main contribution of [2] has been to combine the two approaches by extending Lin and Canny's walk, which is constrained on the surface of the bodies, to a *hierarchical walk*, that can also attempt shortcuts through the inner layers of the Dobkin-Kirkpatrick hierarchy if the coherence is low.

H-Walk is a key yardstick for the ability to adapt to variable coherence, in which respect we find a close correspondence with the purposes of our algorithm. Also the experiments discussed in [2] are interesting for suggesting a criterion to control the coherence level, as well as for comparing the behavior of *H-Walk* and *V-Clip* [37], a more efficient implementation of Lin and Cannys's technique, and showing how *H-Walk* outperforms *V-Clip* for low coherence.

V. Experimental results

In this section we analyze the results of tens of thousands of proximity queries planned to test the behavior of the collision translation algorithms both from scratch (without initialization) and incrementally (with initialization). More specifically, we will consider the following points:

- Trend of the computational costs without initialization.
- Costs of detecting that the polyhedra do not collide.
- Relation between incremental behavior and coherence.
- Comparison with the yardstick algorithms.
- Flexibility under variable coherence.

The input polyhedra are characterized by fairly regular arrangements of vertices on the surface of ellipsoidal shapes, in such a way that almost all the faces are trapezoids (triangles in the case of comparison with PQP). We have also considered two situations: one in which the edges are balanced in length, the other where the faces are very thin and stretched out. The number of vertices of each polyhedron varies from about

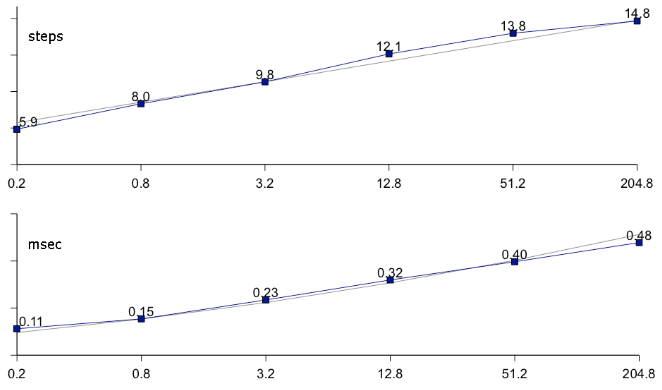

Fig. 11. Trends of the average number of minimization steps (above) and query-time in milliseconds. The gray plots show the approximated $\log$, respectively $\log^2$ functions. Abscissae: thousands of vertices per polyhedron.

200 to about 200,000 and we will refer to such number of vertices as the polyhedron *size*. For any given size, the reported measures are the average of several computations carried out on random general-configuration settings. In particular, while testing the incremental behavior, the choice of the translation direction is unrelated to the motion trajectory.

For ease of reference, we will denote each algorithm by an acronym, namely: *CTA* (Collision Translation Algorithm) for our algorithm, *EGJK* (Extended GJK), *PQP* (Proximity Query Package) and *H-W* (Hierarchical Walk) for those introduced in section IV. The corresponding programs, implemented in the languages Pascal, C and C++, have all been processed with the family of GNU's compilers and run on a Macintosh platform PowerPC G5 (Dual 1.8 GHz, 768 MB RAM).

A. Computations of collision translations from scratch

A first set of experiments was aimed at testing the trend of the computational costs while increasing the size of the polyhedra. As we can see in figure 11, the average number of minimization steps grows as the logarithm of the number of vertices and remains small also in complex cases (less than 15 steps in the average for two polyhedra of about 200,000 faces each). Also the measured query times are in accordance with the estimations and approximate a $\log^2$ trend. The average values reported in figures 11 refer to independent computations of collision translations, carried out without exploiting the spatial coherence. Moreover, a finer analysis shows that the algorithm performs a little worse for thin and stretched faces (20 to 80% increase of the query times).

We have also investigated a little on how *CTA*'s computation time is spent. The collected data are summarized in table I, which should be self-explanatory. The table shows that almost half the time is spent to solve two-dimensional problems for geometric constructions on the section planes, whereas only a negligible fraction of the computation time is required to process and update the polygonal region containing the point of minimum, which is in fact always bounded by few sides. It may also be observed that all the geometric computations can be carried out by using only standard floating-point arithmetic and do not need square root or trigonometric functions.

Types of operations	**Time**
2D geometry on section planes	48%
3D geometry	42%
Indexing to access the representation items	9%
Minimization geometry and control	1%

TABLE I
DISTRIBUTION OF THE COMPUTATION TIMES.

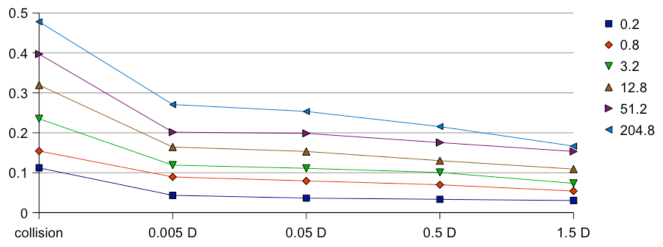

Fig. 12. Average query times (msec) for detecting that the polyhedra do not collide; their minimal separation is represented as a fraction of a reference diameter D. Plot labels: thousands of vertices.

B. Detection that the polyhedra do not collide

In figure 12 we summarize the behavior of the algorithm, again without initialization, when the input polyhedra do not collide. In this case, indeed, the computation times are significantly reduced. The plots in the figure refer to different sizes (see legend) and, for the sake of comparison, start on the left with the query times reported in figure 11. All the other data are for increasing separations (from left to right), the extent of such separations being measured by the minimal distance between the bodies during the motion in direction **d**. In particular, we have considered separations of about $0.005D$ (the bodies get very close to each other), $0.05D$, $0.5D$ and $1.5D$ (the bodies move far away from each other), for a medium diameter D of the polyhedra. The results show that the algorithm runs faster to provide a separation configuration witnessing that the polyhedra cannot collide: the computation times are almost halved even for bodies getting very close to each other and reduce to about $1/3$ when they move far apart.

C. Incremental computation of collision translations

The plots in figure 13 contrast the algorithm's performances *with* and *without* initialization for sequences of 100 configurations that result from sampling at regular intervals a continuous motion of a polyhedron and computing collision translations in an independent direction. In these examples the polyhedra have about 12,800 vertices, whereas the configuration-change parameter δ is $1\%D$ for the upper chart and $0.25\%D$ for the lower one. However, analogous experiments for different polyhedron sizes as well as for sequences resulting from translational, rotational and screw motions show that the situation illustrated in figure 13 is quite typical. In particular, for given polyhedra, the performances do not depend on the type of motion, but on the coherence degree.

The plots of figure 13, where the "floor" lines correspond to computations accomplished in one minimization step, give an intuitive idea of how the algorithm can gradually adapt to

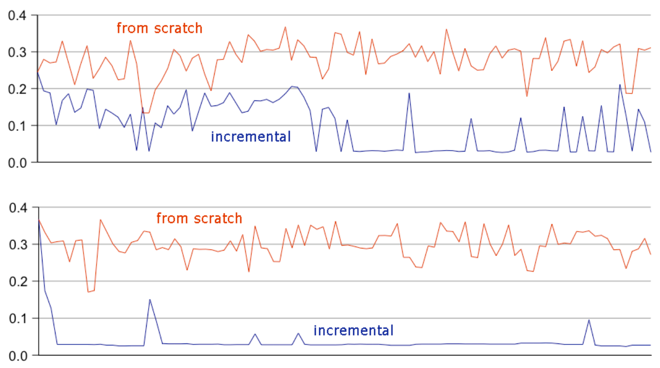

Fig. 13. Typical results for sequences of incremental computations; the average query times of incremental computations (lower plots, in milliseconds) are contrasted with the corresponding times of computations from scratch. The broken lines interpolate 100 samples.

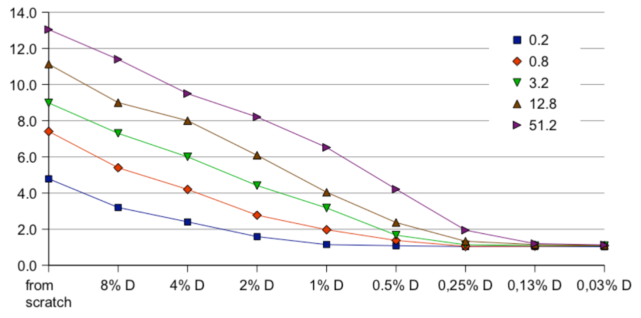

Fig. 14. Summary of the experiments addressing the incremental behavior, where the performances are measured in terms of minimization steps; the plots are labeled with different polyedron sizes (thousands of vertices). Abscissae: values of δ represented as a fraction of a reference diameter D.

varying degrees of coherence between subsequent configurations. Figure 14 shows a clear representation of this behavior, by summarizing the average number of minimization steps for different polyhedron sizes and for different values of δ, where the latter are reported in abscissa as a fraction of a reference diameter D of the input polyhedra. As we can see, the trends are in good accordance with our estimate of section III-D. The plots interpolating the query times would be analogous.

D. Performance comparisons

In order to try a meaningful comparison with *EGJK* and *PQP*, we have run the algorithms on exactly the same settings (i.e., same pairs of polyhedra in the same configurations), where of course the input direction **d** is only relevant for computing collision translations. Notice that the comparison also makes sense when the polyhedra do not collide by translation in the direction **d**, since in that case our algorithm reports suitable information on the separation of the bodies.

About 1,200 pseudo-random settings have been tested to contrast the performance trends of computations from scratch while increasing the size of the polyhedra. The resulting trends for the case of *balanced* edge lengths are drawn in figure 15, where the x- and y-axes report in logarithmic scale the size of polyhedra and the average query times, respectively.

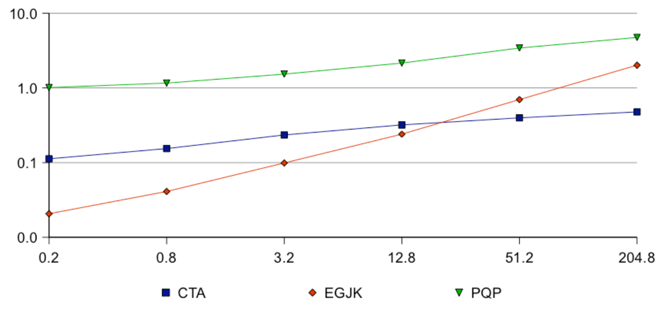

Fig. 15. Trend of the performances for “balanced” edge lengths. Abscissae: thousands of vertices per polyhedron; ordinates: average query time in msec.

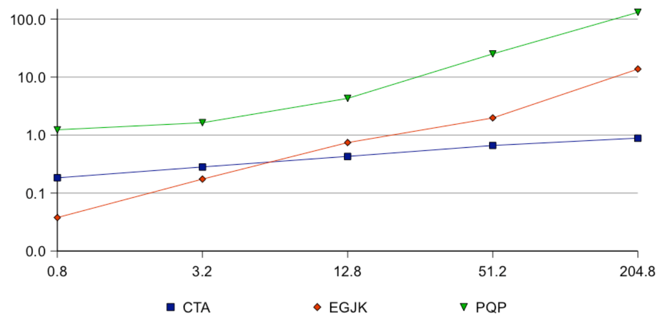

Fig. 16. Trend of the performances for “thin and stretched” faces. Abscissae: thousands of vertices per polyhedron; ordinates: average query time in msec.

As expected, the cost of the computations of *EGJK* grows linearly with the size of the polyhedra, whereas *CTA* and *PQP* seem to show a sublinear trend, again in accordance with the theoretical estimates in the case of *CTA*. In this kind of situations, the query times of *CTA* are the lowest when the polyhedra have about 20,000 vertices or more. Within the considered complexity range, the ratio of the average query times $qt(EGJK)/qt(CTA)$ increases from about 1/5 to about 4. The ratio $qt(PQP)/qt(CTA)$ oscillates in a band between 6.5 and 10 for trapezoidal faces and around 5.5 for triangular faces (trapezoidal faces are indeed a little unfavorable to *PQP* since they can only be represented by couples of triangles).

It may be worth observing that our results are qualitatively and numerically different in the case of *thin-and-stretched* faces, as shown in figure 16. Independently of the approach, the computations turn out to be more expensive and apparently the performances of *PQP* do no longer follow a sublinear trend. With this type of polyhedra the query-time ratio $qt(EGJK)/qt(CTA)$ raises from about 1/5 to about 14, whereas the ratio $qt(PQP)/qt(CTA)$ jumps to a factor of over 100. Furthermore, and quite unexpectedly, the query times of *CTA* are the lowest also in the situations where the polyhedra do not collide but get very close to each other (minimal distance less than 1% of a medium diameter), as illustrated by the chart in figure 17.

A final set of experiments was meant to contrast the incremental performances of *CTA* and *EGJK*. The algorithms have been tested on more than 60 sequences of 100 configurations, where every next configuration is obtained by a short translation and/or a small rotation of a polyhedron. The results

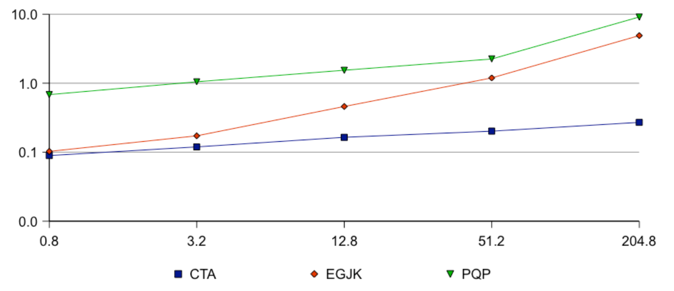

Fig. 17. Trend of the performances when the polyhedra do not collide by translation, but move very closed to each other. Abscissae: thousands of vertices per polyhedron; ordinates: average query time in msec.

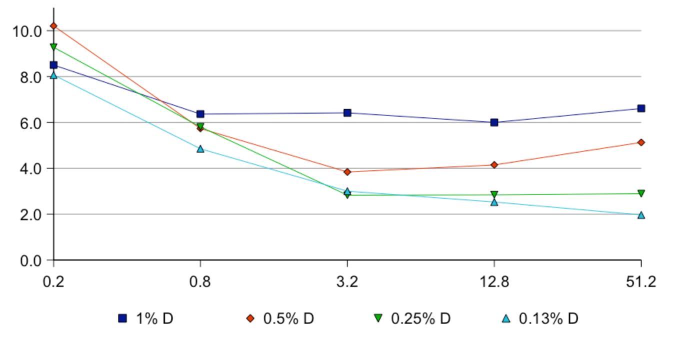

Fig. 18. Trends of incremental performance ratios: the four plots are for different values of δ (see legend). Abscissae: thousands of vertices per polyhedron; ordinates: average query-time ratio $qt(CTA)/qt(EGJK)$.

of these experiments did not reveal any significant difference between the cases of balanced edge lengths and of thin-and-stretched faces, but the only parameter which turns out to affect the query-times rates is δ. The outcomes of the bulk of the experiments on incremental computations are summarized in figure 18, where the x- and y-axes report the size of polyhedra and the average query-time ratio $qt(CTA)/qt(EGJK)$. The four plots, from top to bottom on the right side of the chart, are relative to increasing coherence, i.e. δ is about 1%, 0.5%, 0.25% and 0.13%, respectively, of a medium diameter of the polyhedra. As we can see, the query-time ratios vary from about 10 to 2 and keep favorable to *EGJK*, but decrease for shorter incremental changes and for more complex polyhedra.

E. Behavior under variable coherence

A set of experiments presented in [2] characterize *H-W*'s behavior for variable levels of coherence. Basically, the testing scheme uses a pair of spherical polyhedra, one of which rotates and orbits around the other, under two independent control parameters: the angular rotation step ω between subsequent runs of the algorithm, and the layer l in the Dobkin-Kirkpatrick hierarchy [14] where the initialization features are picked. ω controls the degree of coherence: higher values in the range $[0, 180]$ correspond to lower coherence; l allows a gradual tuning from strong (outermost layer) to weak initialization (innermost layer, as for the original algorithm [14] without initialization). Interestingly, Guibas and colleagues chose to measure the performances, independently of any particular platform, in terms of steps walked per run.

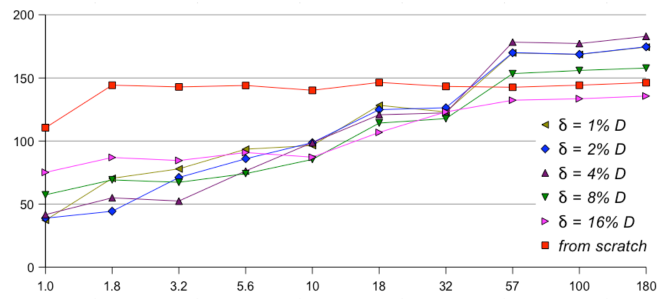

Fig. 19. *CTA*'s behavior for different degrees of coherence and for different choices of δ; polyhedra of 800 vertices. Abscissae: coherence parameter ω (angular rotation step in degrees); ordinates: average number of iterations; labels: initialization parameter δ (D is the diameter of the bodies).

We have arranged for a related set of experiments to test *CTA*'s behavior under (as far as possible) similar conditions. In fact, as a consequence of the structural differences between *H-W* and *CTA*, there are three specific points to consider: (i) as usual for the comparisons with distance algorithms, it is necessary to introduce a direction vector, and we did so in such a way that the polyhedra do always collide by translation; (ii) the control parameter l has been replaced by *CTA*'s δ, a reasonable choice since it realizes a fine tuning from strong (small values) to weak initialization (δ of the order of the diameter D of the bodies); (iii) in the case of *CTA*, an accurate performance measure is the total number of iterations relative to any operation that may be repeated (i.e., outer minimization steps + all inner binary search steps). We ran several thousands of tests in this way, mostly for polyhedra with size 800 and 3,200 that are also used in [2]. The results are summarized in figure 19 for polyhedra of about 800 vertices, the other cases being analogous. Unlike the experiments discussed in subsection V-C and plotted in figure 14, notice that here δ is unrelated to the coherence degree. Furthermore, as in [2], we have analyzed the standard deviation.

Of course, it wouldn't make sense to compare directly the numbers of *H-W*'s walk steps and *CTA*'s iterations. What is interesting to see is how the performance measures vary, relative to the changes of the control parameters. On the one hand, *H-W* and *CTA*'s trends share some qualitative features: the initialization parameters clearly affect the performances when the coherence is very high or very low; for $\omega > 30$ degrees, strong initialization either does not significantly improve or worsens the performances. On the other hand, two main differences emerge from the analysis of the results: First, *CTA* seems to adapt better than *H-W* to *low* coherence, in the sense that the relative worsening of *CTA*'s performance while changing the initialization parameter is more moderate: about 35% against 65% taken from the plot in [2] for 800 vertices; 25% against 80% for 3,200 vertices. This means, for instance, that a suboptimal choice of the initialization parameter would result into a slightly smoother behavior in the case of *CTA*. Moreover, it is easy to dynamically update the value of δ on the basis of the configuration change. Second, the ratio between the standard deviation and the corresponding performance measure is favorable to *H-W*: about 14% against 70% for high coherence; 4% against 35% for low coherence. In other words, there is more dispersion among the costs of individual runs of *CTA*, which means higher levels of uncertainty to predict the time required by a single computation; as pointed out in [2], this may be a relevant issue for time-critical planning.

F. Discussion

Based on the above results, the most remarkable feature of our algorithm is the low rate of growth of the response time for increasing complexities of the polyhedra. This feature is particularly manifest for computations from scratch, but also emerges from the trends relative to the incremental tests, although the query times of *EGJK* have been systematically shorter than those required by *CTA*. A possible explanation of the latter phenomenon is that *CTA*, in its present form, does not exploit any initialization information to solve instances of the collision translation subproblem for pairs of planar sections, whose cost is $\Theta(\log n)$, and that at least one such instance must always be solved. Thus, *CTA*'s costs have a logarithmic lower bound, whereas such algorithms as *EGJK* or *V-Clip* [37], as well as *H-W* with strong initialization, are expected to run in nearly constant time when the coherence is high.

As far as the incremental behavior is concerned, our interpretation of the experimental results is that the algorithm is not as fast as it could perhaps be, and some refinements in this respect may be worth further study. However, in some sense the non-optimality of the incremental response can also be ascribed to a necessary tradeoff to guarantee reasonably good performances independently of the coherence level, which may be a desirable feature for applications with bounded tolerance on the response times. Moreover, it should be observed that the gap between *CTA* and *EGJK*'s incremental performances reduces with the size of the polyhedra, and more distinctly for higher coherence (small values of δ). This observation is probably of limited practical relevance, but it is interesting if we consider the intrinsic properties of the algorithm, which are characterized up to constant factors. Finally, *CTA* is less sensitive to variations of the coherence degree. In particular, if the coherence drops down its performances do not deteriorate as dramatically as *EGJK*'s, and the adjustment to coherence fluctuations seems to be slightly smoother also relative to *H-W*.

VI. Conclusions

We have presented an asymptotically fast algorithm for computing collision translations of convex polyhedra, with additional potential for incremental computations. After introducing the peculiar convex minimization approach, we have analyzed the behavior of the algorithm and compared its performances with those of other key algorithms. To this aim, we have considered a variety of settings, including sequences of slightly changing configurations to test the incremental behavior. A major strength of the algorithm, substantiated by the results presented in section V, is the low rate of growth of the response time for increasing complexities of the polyhedra.

We conclude by mentioning some possible directions of future work. Firstly, a refinement of the technique focusing the search for the point of minimum in the vicinity of a previous solution may be worth further attention. More specifically, in the present implementation only a pair of planar sections is passed forward to the next incremental computation, not the actual features answering the proximity query. Thus, the procedure for computing collision translations of two planar sections is invoked at least once, whereas checking the previous closest features or their close neighbors could be enough. Improving this technique may be effective, since several incremental computations are completed in just one minimization step. Another crucial point in order to develop a robust algorithm, which has to be investigated in more depth, is the choice of the switch conditions between general and non-general configurations for applying the technique outlined in section III-B. Finally, as pointed out in [3], other proximity problems can be approached in the same way. This is straightforward for testing intersections and for computing the collision depth in a given direction, but there still remain technical problems to be solved in order to extend the approach to efficiently compute distances between convex polyhedra.

Acknowledgments

This paper builds upon and extends some previous results presented in [42] and [43]. We are grateful to the anonymous reviewers for their valuable comments and suggestions.

References

[1] S. Cameron, “Dealing with geometric complexity in motion planning,” in *IEEE Workshop on Practical Motion Planning in Robotics: Current Approaches and Future Directions*, 1996.

[2] L. J. Guibas, D. Hsu, and L. Zhang, “A hierarchical method for real-time distance computation among moving convex bodies,” *Computational geometry: theory and applications*, vol. 15, no. 1-3, pp. 51–68, 2000.

[3] C. Mirolo, “Convex minimization on a grid and applications,” *Journal of Algorithms*, vol. 26, no. 2, pp. 209–237, 1998.

[4] S. Cameron, “A comparison of two fast algorithms for computing the distance between convex polyhedra,” *IEEE Trans. on Robotics and Automation*, vol. 13, no. 6, pp. 915–920, 1997.

[5] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, “Fast proximity queries with swept sphere volumes,” Department of Computer Science, University of North Carolina, Tech. Rep. TR99-018, 1999.

[6] M. C. Lin and S. Gottschalk, “Collision detection between geometric models: a survey,” in *Proc. of the IMA Conf. on Math. of Surfaces*, 1998.

[7] N. M. Amato, O. B. Bayazit, L. K. Daleand, C. Jones, and D. Vallejo, “Choosing good distance metrics and local planners for probabilistic roadmap methods,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 1998, pp. 630–637.

[8] J. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, “I-collide: An interactive and exact collision detection system for large-scaled environments,” in *Proc. of the ACM Int. 3D Graphics Conf.*, 1995, pp. 189–196.

[9] 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 J. of Robotics and Autom.*, vol. 4, no. 1, pp. 193–203, 1988.

[10] M.-Y. Ju, J.-S. Liu, S.-P. Shian, Y.-R. Chien, K.-S. Hwang, and W.-C. Lee, “A novel collision detection method based on enclosed ellipsoid,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 2001, pp. 2897–2902.

[11] N. K. Sancheti and S. S. Keerthi, “Computation of certain measures of proximity between convex polytopes: A complexity viewpoint,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 1992, pp. 2508–2513.

[12] K. Sridharan and S. S. Keerthi, “Computation of a penetration measure between 3D convex polyhedral objects for collision detection,” *J. of Robotic Systems*, vol. 18, no. 11, pp. 623–631, 2001.

[13] C. Turnbull and S. Cameron, “Computing distances between NURBS-defined convex objects,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 1998, pp. 3685–3690.

[14] D. P. Dobkin and D. G. Kirkpatrick, “Determining the separation of preprocessed polyhedra: A unified approach,” in *Proc. of ICALP*, ser. LNCS 443, 1990, pp. 400–413.

[15] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri, “Computing the intersection-depth of polyhedra,” *Algorithmica*, vol. 99, no. 6, pp. 518–533, 1993.

[16] C. J. Ong, “Properties of penetration between general objects,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 1995, pp. 2293–2298.

[17] P. Jiménez and C. Torras, “Benefits of applicability constraints in decomposition-free interference detection between nonconvex polyhedral models,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 1999, pp. 1856–1862.

[18] F. Thomas, C. Turnbull, L. Ros, and S. Cameron, “Computing signed distances between free-form objects,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2000, pp. 3713–3718.

[19] S. Redon, A. Kheddar, and S. Coquillart, “An algebraic solution to the problem of collision detection for rigid polyhedral objects,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2000, pp. 3733–3738.

[20] P. G. Xavier, “Implicit convex-hull distance of finite-screw-swept volumes,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2002, pp. 847–854.

[21] M. C. Lin and J. Canny, “A fast algorithm for incremental distance calculation,” in *Proc. of the IEEE Intl. Conf. on Robotics and Autom.*, 1991, pp. 1008–1014.

[22] D. E. Johnson and E. Cohen, “Bound coherence for minimum distance computations,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 1999, pp. 1843–1848.

[23] Y. J. Kim, M. C. Lin, and D. Manocha, “Incremental penetration depth estimation between convex politopes using dual-space expansion,” *IEEE Trans. on Visual. and Comp. Graphics*, vol. 10, no. 2, pp. 152–163, 2004.

[24] B. Martinéz-Salvador, A. P. del Pobil, and M. Pérez-Francisco, “A hierarchy of detail for fast collision detection,” in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Sys.*, 2000, pp. 745–750.

[25] C. J. Ong and E. Huang, “An incremental version of growth distance,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 1998, pp. 3671–3677.

[26] C. J. Ong and E. G. Gilbert, “Fast versions of the Gilbert-Johnson-Keerthi distance algorithm: Additional results and comparisons,” *IEEE Trans. on Robotics and Automation*, vol. 17, no. 4, pp. 531–539, 2001.

[27] K. Sundaraj, D. d’Aulignac, and E. Mazer, “A new algorithm for computing minimum distance,” in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Sys.*, 2000, pp. 2115–2120.

[28] L. J. Guibas, F. Xie, and L. Zhang, “Kinetic collision detection: Algorithms and experiments,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2001, pp. 2903–2910.

[29] E. J. Bernabeu, J. Tornero, and M. Tomizuka, “Collision prediction and avoidance amidst moving objects for trajectory planning applications,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 2001, pp. 3801–3806.

[30] S. A. Ehmann and M. C. Lin, “Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching,” in *Proc. of the IEEE-RSJ Int. Conf. on Intell. Robots and Sys.*, 2000, pp. 2101–2106.

[31] P. M. Hubbard, “Approximating polyhedra with spheres for time-critical collision detection,” *ACM Trans. on Graphics*, vol. 15, no. 3, pp. 179–210, 1996.

[32] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, “Fast distance queries with rectangular swept volumes,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2000, pp. 3719–3726.

[33] Y. Sato, M. Hirata, T. Maruyama, and Y. Arita, “Efficient collision detection using fast distance-calculation algorithms for convex and nonconvex objects,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 1996, pp. 771–778.

[34] T. Siméon, J. P. Laumond, C. V. Geem, and J. Cortes, “Computer aided motion: Move3D with MOLOG,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2001, pp. 1494–1499.

[35] P. G. Xavier, “Fast swept-volume distance for robust collision detection,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 1997, pp. 1162–1169.

[36] G. Zachmann, “Minimal hierarchical collision detection,” in *Proc. of the ACM Symp. on Virtual Reality Software and Techn.*, 2002, pp. 121–128.

[37] B. Mirtich, “V-clip: Fast and robust polyhedral collision detection,” *ACM Trans. on Graphics*, vol. 17, no. 3, pp. 177–208, 1998.

[38] M. Reggiani, M. Mazzoli, and S. Caselli, “An experimental evaluation of collision detection packages for robot motion planning,” in *Proc. of the IEEE/RSJ Int. Conf. on Intell. Robots and Sys.*, 2002, pp. 2329–2334.

[39] A. S. Nemirovsky and D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

[40] B. Chazelle and D. P. Dobkin, “Intersection of convex objects in two and three dimensions,” *J. of the ACM*, vol. 34, no. 1, pp. 1–27, 1987.

[41] C. Mirolo, “Polylogarithmic algorithms for collision detection based on convex minimization,” Dip. di Matematica e Informatica dell’Univ. di Udine, Tech. Rep. UDMI/01/94/RR, 1994.

[42] C. Mirolo and E. Pagello, “Flexible exploitation of space coherence to detect collisions of convex polyhedra,” in *Proc. of the IEEE Int. Conf. on Robotics and Autom.*, 2001, pp. 3783–3788.

[43] S. Carpin, C. Mirolo, and E. Pagello, “A performance comparison of three algorithms for proximity queries relative to convex polyhedra,” in *Proc. IEEE Int. Conf. on Robotics and Autom.*, 2006, pp. 3023–3028.