A Performance Comparison of Three Algorithms for Proximity Queries relative to Convex Polyhedra

Stefano Carpin
School of Engineering and Science
International University Bremen
Bremen, Germany
Email: s.carpin@iu-bremen.de

Claudio Mirolo
Dept. of Mathematics and Informatics
University of Udine
Udine, Italy
Email: claudio@dimi.uniud.it

Enrico Pagello
Dept. of Information Engineering
University of Padova
Padova, Italy
Email: epv@dei.unipd.it

***Abstract*— This paper presents a comparative analysis relative to the experimental performances of an asymptotically fast and incremental algorithm, recently developed to compute collision translations for pairs of convex polyhedra. The algorithm may be worth considering because it solves a proximity problem which is less widely addressed than distance, as well as because of its peculiar computation strategy, well suited to work without initialization, but also endowed with an inherently embedded mechanism to exploit spatial coherence. Numerical data characterizing the behavior of the algorithm with respect to the complexity of the polyedra have already been discussed elsewhere, thus here the main focus is on contrasting its performances with those of two popular algorithms designed to compute distances between polyhedra. Although the considered "yardsticks" answer different proximity queries, and although one of the techniques is meant to deal with general polyhedra, the results presented in this paper should help to assess the efficacy and potential of the approach under analysis. All the three algorithms, indeed, share the same kind of application context; moreover, on the basis of the asymptotic bounds discussed in the literature, distances and collision translations require similar computational efforts. A thorough comparison of the reported query times and, more significantly, of the corresponding trends seems to show that the behavior of the novel algorithm is quite interesting, especially when used without initialization, what should encourage further work on its peculiar approach.**

I. Introduction

In this paper we analyze 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. The key idea characterizing our approach is that the computation of collision translations for two convex bodies can be reduced to computation of collision translations for pairs of planar sections and minimization of a bivariate convex function.

This idea can be exploited to design an algorithm running in $O(\log^2 n)$ average time [1], where n is the total number of vertices, an asymptotic trend which meets that of the best algorithms proposed to answer similar proximity queries. This complexity bound refers to computations *from scratch*, i.e., without initialization, when no previous proximity information is exploited. However, it is quite typical of on-line motion planning, simulation, animation and CAD problems 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 queries are crucial, and exploiting the *spatial* and *temporal coherence* can significantly speed up the computation. Efficient algorithms designed to this purpose are referred to as *incremental* algorithms.

The algorithm is worth of interest for two main reasons. On the one hand, the specific proximity problem we consider is not as widely addressed as that of distance determination. On the other hand, the core approach is quite peculiar, suitable both to work without initialization (computations *from scratch*) and to exploit coherence (*incremental* computations) with the aid of a mechanism that is inherently embedded in the computation strategy. The results of various numerical experiments have already been the subject of some previous work [2], [3] aimed at characterizing the computational costs in terms of number of minimization steps and trend of response times, as well as to estimate the speedup attainable with the incremental strategy. As yet, however, there were no experimental data for a comparison with other proposals, what would allow a more comprehensive understanding of the features of our algorithm. This is why the focus of this paper is precisely on contrasting its performances with those of two well known algorithms designed to compute distances between polyhedra [4], [5].

Although the "yardstick" algorithms are not equivalent, the results presented in this paper are useful for a first assessment of the efficacy and potential of the novel approach. All the three algorithms, indeed, share the same kind of application frameworks, and answering distance and collision translation queries appear to require a similar computational effort, as 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 this kind of queries is generally thought to be critical to the development of effective tools for motion planning, as pointed out e.g. in [7], and for a variety of applications in other fields where the geometry plays an

important role [8]. Several algorithms apply to convex models, e.g. [9], [10]; in the case of convex polytopes the best solutions of various proximity problems exploit the properties of hierarchical representations and require $O(\log^2 n)$ worst-case time [11]. Some efforts, however, have also been addressed to more general bodies, e.g. [12], [13].

Since the work by Lin and Canny in the early '90s [14], the need to deal with complex settings has fostered research on incremental algorithms, e.g. [4], [15], [16], [17], among which it is interesting to mention *H-Walk* for the capability to adapt to variable coherence [18]. Another crucial point is the design of suitable spatial representations to speed up the broad phase, which is aimed at selecting few pairs of primitive volumes (e.g., convex polyhedra) for the specific proximity tests. A representative example of this approach is the application of *kinetic data structures* [19]. Other hierarchical representations based on bounding volumes of simple shape are considered in several papers, e.g. [8], [20], [21].

Two algorithms are of specific relevance for the analysis presented here, namely the enhancement of GJK proposed by Cameron [4] and the distance computation procedure of the Proximity Query Package developed by Larsen, Gottschalk, Lin and Manocha [5]. It is quite natural to choose these algorithms as benchmarks for proximity problems since they are widely known and have already been compared with other techniques, e.g. [4], [17], [22], [23].

Organization of the paper

In section II we outline the algorithm for computing collision translations and its mechanism to exploit coherence. In section III we introduce the "yardstick" algorithms for the performance comparison and we motivate the choice. Finally, the experimental results are summarized in section IV.

II. The collision translation algorithm

The algorithm is based on some previous work on a convex-minimization approach to determine collision translations [1] and applies the refinements described in [2], [3].

A. Collision translations and convex minimization

Collision translations for two convex bodies P and Q can be reduced to minimization of a bivariate convex function that represents collision translations relative to pairs of planar sections of P and Q. More formally, we can prove the following proposition [1]:

> Let P, Q be two closed 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 a 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 $dist(\{p + \tau d \mid p \in X\}, Y) = 0$; $\rho(x)$ 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. Clearly, if a collision translation for P and Q is defined, then its extent is the minimum of φ:

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

and the planar sections corresponding to the point of minimum contain the contact points.

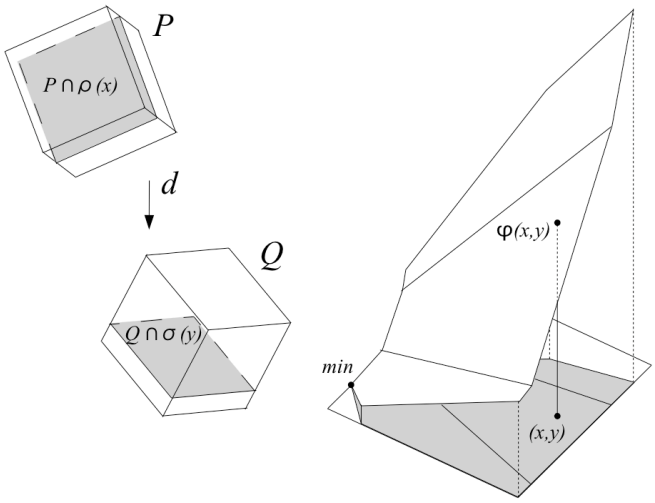

Fig. 1. Relationship between collision translations and convex minimization.

The meaning of the proposition is also 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 relative to two convex polyhedra by standard convex minimization [24]. The next step is to transform a search problem on a continuous domain into a search problem on a discrete domain. In [1] the discrete search domain is the set of rectangles of an isothetic grid. Under reasonable assumptions, the grid rectangle containing the point of minimum can be found in $O(\log n)$ minimization steps in the average, where n 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. Improved convex minimization

A deeper characterization of the properties of the convex function φ is introduced in [2]. Since P and Q are polyhedra, φ's graph is also faceted and its topology projects into a corresponding polygonal partition of φ's domain. Such a partition is a more appropriate discrete structure to search for the point of minimum. For our purposes, however, it is necessary to extend the partition outside φ's domain, to cover the whole rectangle $\{(x, y)|P \cap \rho(x) \neq \emptyset \wedge Q \cap \sigma(y) \neq \emptyset\}$, representing all possible pairs of planar sections of P and Q. It is possible to achieve this goal by introducing suitable invariant

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

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. 2. Structure of the collision translation algorithm. For simplicity, the output refers only to the situations where a collision translation is defined.

properties characterizing the orientation of the *cut lines* built in the minimization process, i.e., straight lines perpendicular to φ's gradient, if drawn through points within the domain, and straight lines which do not intersect φ's domain, if drawn through points outside the domain (notice that in a standard minimization technique, such as the method of centers of gravity, a convex region containing the point of minimum is repeatedly shrunk by cutting off a slice at each step with a cut line through the centroid). In the following, we will refer to the complete partition mentioned above as the *mark of* φ.

By definition, the orientation of a cut line does not change if the cut point moves within the same cell of the mark. 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 detect collisions of the pair of planar sections $P \cap \rho(x)$ and $Q \cap \sigma(y)$. Given the information on either the contact or the separation of the planar sections, we can build a cut line through (x, y) and, in addition, we can find a more favorable point of the corresponding cell of the mark, i.e., the point such that the cut line 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. A convenient characterization of the mark distinguishes three types of cells, relative to the outcomes of the collision translation algorithm for pairs of planar sections: (a) *edge contact*, if a pair of edges of the planar sections get into contact in a collision configuration; (b) *vertex contact*, if a vertex "hits" the interior of a polygonal section in a collision configuration; (c) *separation configuration*, if the planar sections do not collide.

Figure 2 sketches the algorithm designed to compute collision translations for two polyhedra P, Q and a translation direction d. At line (1) the region M is initialized 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. We can reasonably estimate 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. In [2] it is also argued that $k = O(\log n)$ and that, after considering all the subproblems, the algorithm runs in $O(\log^2 n)$ time in the average.

C. Exploiting spatial coherence

For a variety of applications a proximity query needs to be repeatedly asked after short intervals of time, so that two subsequent answers are expected to be closely related to each other. In similar situations considerable speedup may be gained by exploiting the information on the previous proximity computation. As said before, algorithms designed to this purpose are referred to as *incremental* algorithms in the literature. The very nature of our approach makes it possible to endow the 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 happens to lie outside of it.

As to the first problem, the chosen 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 (positions and orientations of P, Q and d; 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 this neighborhood and a mechanism for switching to the standard search strategy must be provided. A possible technique works as follows. The search starts at the point p representing the previous solution, but the minimization region M is initialized as in the *from-scratch* case (figure 2). 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 pc. 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.

The incremental behavior of the algorithm is achieved simply by the operations in lines (2) and (9) of figure 2: initially (2) the previous solution is chosen as first cut point and at the end of each iteration (9) the cut point is chosen as sketched above. It is worth observing that the search focus can be tuned by means of the parameter δ, which depends on the actual change of the relative configuration of P, Q and d.

As a consequence, 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. Call θ the ratio between the length δ, measuring the configuration change, and the initial height of the minimization region M (for simplicity, suppose that M approximates a square). The expected number of cells intersecting the neighborhood $N^\delta(p)$ over the total number C of cells is about $Area(N^\delta(p))/Area(M) = 4\theta^2$ and then the number k of iteration steps is proportional to $\log C - 2\log(1/\theta)$ So, if the updated solution lies inside $N^\delta(p)$, the gain with respect to the standard strategy is of about $O(\log(1/\theta))$ iterations in the average. Since this rough estimate appears to be in good accordance with the experimental trends [3], this means that the updated solution falls inside $N^\delta(p)$ with *high* probability and witnesses the important role which coherence may play.

III. Enhanced GJK and PQP

We now introduce the "yardstick" algorithms that we have considered for the performance comparison: the extended GJK [4] and the distance computation procedure available in the Proximity Query Package [5]. As said before, the comparison is not completely fair since these algorithms answer different proximity queries, being designed to compute distances rather than collision translations. Moreover, for PQP it is also the generality of the algorithm that should be taken into account, since it is not restricted to convex bodies, but in this case we are also interested in achieving a better understanding of the power of hierarchical structures as opposed to convexity properties. Once this is clear, we think that this kind of comparison can be useful for a first appraisal and will hopefully suggest possible directions of future work.

A. Enhanced GJK: convex polyhedra and spatial coherence

In [4] Cameron proposes an enhancement of the classical Gilbert, Johnson and Keerthi's (GJK) algorithm [9]. The original GJK algorithm computes the distance between two convex polyhedra by finding the distance from the origin of their Minkowski difference, which is a convex set as well and represents the separation of the given bodies. The algorithm constructs and maintains a *simplex*, built from linearly independent vertices of the given polyhedra, and iteratively checks whether it is optimal, i.e. whether it minimizes the distance; if not, the simplex is updated. The strength of the original algorithm is the efficiency of these operations.

Cameron has shown that this approach can be improved by applying a *hill climbing* technique while looking for a better simplex. His 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 relative configuration arise. Moreover, Cameron argues that under these hypotheses the behavior of his refined algorithm tends to be similar to the behavior of the incremental technique proposed by Lin and Canny [14].

The reason for considering the enhanced GJK algorithm is twofold. On the one hand, it is very fast in practice, and under similar conditions with respect to the algorithm for computing collision translations: the input bodies are indeed restricted to pairs of *convex* polyhedra and it is well suited for exploiting spatial coherence. On the other hand, plenty of experimental data are already available which compare variants of the GJK scheme, including the enhanced GJK, with a variety of other approaches to distance computation [4], [22], [17].

B. PQP: general polyhedra and hierarchical structures

The PQP library implements a set of algorithms discussed in [5]. More specifically, it can answer three types of queries: collision (interpenetration) detection, exact separation distance and approximated separation distance. PQP exploits a *bounding volume hierarchy* in order to speed up the computation: each object is bounded by a certain shape for which the query at hand can be easily answered. The bounding volumes are organized as hierarchical structures, usually trees, in such a way that volumes found at deeper levels are smaller and approximate better the actual shape of (parts of) the object; eventually, the leaves of such structures represent exactly the components of the objects. Different kinds of bounding volumes have been considered in the literature. PQP exploits oriented bounding boxes for collision detection and swept spheres for distance computation.

In order to understand correctly the results discussed in this paper, it is important to notice 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 step, and then becomes ready to answer multiple queries. Moreover, it may be worth noticing that PQP does not exploit spatial coherence; in other words, each computation restarts from scratch as if the query were the first one. However, since PQP is widely known, it is a natural candidate benchmark in the field of proximity algorithms. Also in this case the results of experimental comparisons with related tools are available, as e.g. in [23], where the authors argue that in most situations PQP's performances are comparable to the fastest algorithms. Another reason for considering PQP lies on its use of hierarchical structures and bounding volumes, which makes a comparison of the performance trends attainable with this and other approaches interesting in itself.

IV. Experimental results

In this section we summarize a detailed analysis of the results of several thousands of proximity queries planned in order to test the properties of the algorithms both *from scratch* (without initialization) and *incrementally* (with initialization). In order to try a meaningful comparison, we have decided to run the algorithms on exactly the same settings (i.e., same pairs of polyhedra in the same configurations), where of course the motion direction was only relevant for computing distance translations. It may also be noticed that the comparison makes still sense even if the polyhedra do not collide by moving

one of them in the given direction, since in that case the algorithm outlined in section II reports suitable information on the separation of the bodies (a pair of witness points proving the separation for translations in the given direction).

The input polyhedra are characterized by fairly regular arrangements of vertices on the surface of ellipsoidal shapes, in such a way that either almost all the faces are trapezoids or all are triangles. For both trapezoidal and triangular faces, 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 200 to about 200,000. We will refer to such number of vertices as to the polyhedron *size*, and whenever we speak about query times (*qt*) we always mean the average query time of several computations carried out on different settings where the polyhedra have the same size.

For ease of reference, we denote each algorithm by a suitable acronym, namely: *CTA* (Collision Translation Algorithm) for the algorithm outlined in section II, *EGJK* (Extended GJK) and *PQP* (Proximity Query Package) for those introduced in section III. 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 from scratch

A first set of experiments is based on about 1,200 different settings, generated randomly, and was meant to contrast the performance trends relative to computations from scratch (without initialization) while increasing the size of the polyhedra. The resulting trends for the case of "balanced" edge lengths can be seen in figure 3, where the x- and y-axes report in logarithmic scale the size of polyhedra (thousands of vertices) and the measured qts (milliseconds), respectively. 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* become 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 triangles.

It may be worth observing that the results are different in the case of "thin-and-stretched" faces, as shown in figure 3. Independently of the approach, this case turns 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 over 100. Furthermore, and quite unexpectedly, the qts of *CTA* are the lowest when 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 4.

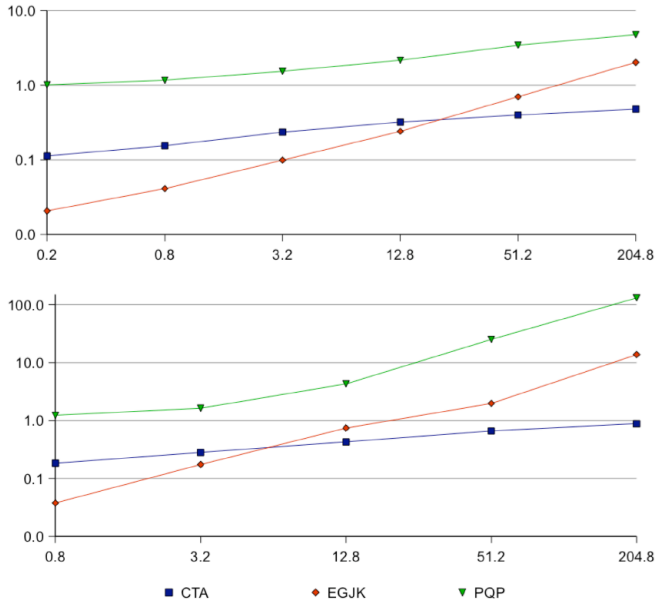

Fig. 3. Trend of the performances for balanced edge lengths (above) and for thin and stretched faces (below). Abscissae: thousands of vertices per polyhedron; ordinates: qt in msec.

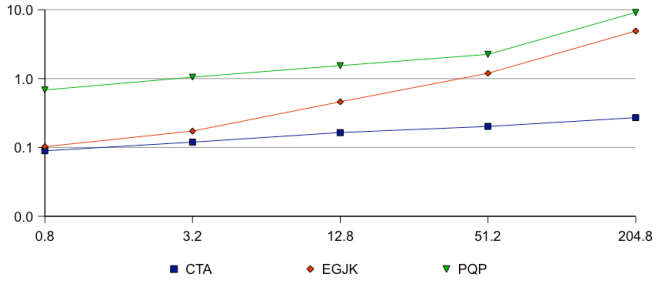

Fig. 4. Trend of the performances when the polyhedra do not collide by translation, but move very closed to each other.

B. Incremental Computations

Another set of experiments was meant to contrast the incremental performances of *CTA* and *EGJK*, based on more than 60 sequences of 100 slightly changing configurations. In such sequences every next arrangement is obtained by a short translation and/or a small rotation of a polyhedron. In these experiments, no significant difference has been revealed between the cases of "balanced" edge lengths and of "thin-and-stretched" faces, but the only parameter which turned out to have affected the query-times rates is the measure δ of the configuration (see section II-C).

The outcomes of the bulk of the experiments on incremental computations are summarized in figure 5, where the x- and y-axes report the size of polyhedra (thousands of vertices) and the average query-time ratio $qt(CTA)/qt(PQP)$. The four plots, from top to bottom on the right side of the chart, are relative to decreasing values of δ (about 1%, 0.5%, 0.25% and 0.13%, respectively, of a medium diameter of the polyhedra: the qt rates decrease for shorter incremental changes). In the

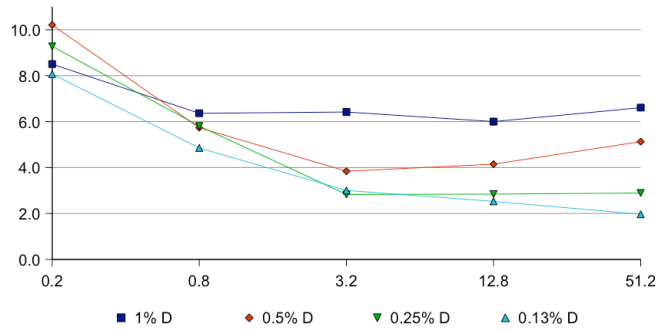

Fig. 5. Trends of incremental performance rates: the plots are relative to different values of δ. Abscissae: thousands of vertices per polyhedron; ordinates: average of *qt(CTA)/qt(EGJK)*.

considered settings for testing incremental computations, the query times of *EGJK* have been systematically lower than those required by *CTA*, but it may also be observed that the gap always reduces with the size of the polyhedra, more distinctly for small values of δ . The reported query-time ratios vary from about 10 to about 2, where the factor 2 applies to the case of small δ (about 0.13% of a medium diameter of the bodies) and complex polyhedra (about 50,000 vertices).

V. Conclusions

The main focus of this paper has been on an experimental comparison of the performances relative to an algorithm for computing collision translations of convex polyhedra and two well known algorithms for computing distances: extended GJK and PQP. As already pointed out, in order to interpret the results of the comparison it is important to take into account the diversity of scope and features of the techniques, in particular that PQP applies to general polyhedra.

Based on the results presented in section IV, the most remarkable feature of our algorithm for computing collision translations is the low rate of growth of the response time with respect to an increasing number of vertices of the given polyhedra. This feature is particularly manifest for computations from scratch, as shown by the plots referred to in section IV-A, but also emerges from the trends relative to the incremental tests summarized in figure 5. However, 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 a refinement of the technique focusing the search for the point of minimum in the vicinity of a previous solution may be worth some further work. More specifically, in the present implementation only a pair of planar sections is passed forward to the next (incremental) computation, and not the actual items answering the proximity query, thus at least one application of the procedure for answering the proximity query for two polygons is required, whereas checking those items or their close neighbors could be enough. Indeed, such an improvement may be effective since several incremental computations are completed in just one minimization step, as reported in [3].

References

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

[2] C. Mirolo and E. Pagello, "Fast convex minimization to detect collisions between polyhedra," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 2000.

[3] ——, "Flexible exploitation of space coherence to detect collisions of convex polyhedra," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2001.

[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, 1997.

[5] E. Larsen, S. Gottshalck, M. 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. Lin and S. Gottschalk, "Collision detection between geometric models: a survey," in *Proc. of the IMA Conf. on Mathematics of Surfaces*, 1998.

[7] N. Amato, O. Bayazit, L. D. 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 Automation*, 1998, pp. 630–637.

[8] J. Cohen, M. 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 Conference*, 1995, pp. 189–196.

[9] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. of Robotics and Automation*, vol. 4, no. 1, pp. 193–203, 1998.

[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. of the IEEE International Conference on Robotics and Automation*, 2001, pp. 2897–2902.

[11] D. Dobkin and D. Kirkpatrick, "Determining the separation of preprocessed polyhedra - a unified approach," in *Proc. of ICALP*, ser. LNCS 443, 1990, pp. 400–413.

[12] C. J. Ong, "Properties of penetration between general objects," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1995, pp. 2293–2298.

[13] 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 Automation*, 2000, pp. 3713–3718.

[14] M. Lin and J. Canny, "A fast algorithm for incremental distance calculation," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation,*, 1991, pp. 1008–1014.

[15] D. Johnson and E. Cohen, "A framework for efficient minimum distance computations," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 3678–3684.

[16] Y. Kim, M. Lin, and D. Manocha, "Incremental penetration depth estimation between convex politopes using dual-space expansion," *IEEE transactions on visualization and computer graphics*, vol. 10, no. 2, pp. 152–163, 2004.

[17] C. Ong and E. Gilbert, "Fast versions of the gilbert-johnson-keerthi distance algorithm: additional results and comparisons," *IEEE transactions on robotics and automation*, vol. 17, no. 4, pp. 531–539, 2001.

[18] L. 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.

[19] L. Guibas, F. Xie, and L. Zhang, "Kinetic collision detection: Algorithms and experiments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 2903–2910.

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

[21] G. Zachmann, "Minimal hierarchical collision detection," in *Proc. of the ACM Symposium on virtual reality software and technology*, 2002, pp. 121–128.

[22] B. Mirtich, "Fast and robust collision detection," *ACM transactions and graphics*, vol. 17, no. 3, pp. 177–208, 1998.

[23] 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 Intelligent Robots and Systems*, 2002, pp. 2229–2334.

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