0.0in

Benefits of Applicability Constraints in Decomposition-free Interference Detection between Nonconvex Polyhedral Models [*]

P. Jiménez C. Torras

Institut de Robòtica i Informàtica Industrial (CSIC - UPC)
Gran Capità 2-4 (Edifici NEXUS), E-08034 Barcelona, Spain
e-mail: *pjimenez@iri.upc.es, ctorras@iri.upc.es*

Abstract

Nonconvex polyhedral models of workpieces or robot parts can be directly tested for interference, without resorting to a previous decomposition into convex entities [15]. Here we show that this interference detection, based on the elemental edge - face intersection test, can be performed efficiently: a strategy based on applicability constraints reduces drastically the set of edge - face pairings that have to be considered for intersection. This is accomplished by using an appropriate representation, the Spherical Face Orientation Graph, developed by the authors, as well as feature pairing algorithms, based on the plane sweep paradigm, that have been adapted to work on that representation. Furthermore, the benefits of such a strategy extend to the computation of a lower distance bound between the polyhedra, both lowering the computational effort and improving the quality of the bound. Experimental results confirm the expected advantages of this strategy.

1 Introduction

The execution time of an interference detection algorithm is obviously very tightly related to the complexity of the object models involved. If polyhedral models are used, and this is the most popular option, interference is checked by means of a number of very simple edge-face tests. In other words, the formulation is very easy from an algebraic point of view, but may be combinatorially expensive if the description of the object requires a large number of primitives. This is the case of objects with complicated shapes for which interference needs to be assessed exactly, in applications such as motion in contact, or assembly/disassembly problems. Therefore, techniques that allow to perform interference detection with the fewest number of elemental descriptors are welcome. Here, one such complexity-lowering technique is presented.

The first reduction of descriptors comes from the adoption of a decomposition-free approach to interference detection [15]. The problem is not the decomposition process per se, which in general has to be performed only once, but rather its result: the introduction of a large number of fictitious primitives, that have to be considered in the interference tests, although they do not correspond to actual features of the objects. This is avoided if no such decomposition is needed.

A further reduction in the number of polyhedral primitives to be considered can be attained by restricting as much as possible the parts of the polyhedra where the interference detection test must be applied. Among the different strategies devised to this end, we have to mention

- the *incremental minimum distance realization* technique [13] used in the collision detection library *I_COLLIDE* [6], where a neighborhood criterion is exploited together with coherence between time frames; originally relying on the convexity of the involved polyhedra, this technique has recently been extended to cope with nonconvex polyhedra [14]);
- the *exploitation of hierarchical boundary representations* [8], that allows a rapid focusing on the regions of the objects boundaries relevant to detect collisions, as shown in the *RAPID* and *V_COLLIDE* libraries [9];
- *back-face culling* [16], where faces whose normal has a negative projection on the relative motion vector are eliminated;
- the *separating vector* algorithm used to develop *Q_COLLIDE* [5], which combines an ef-

[*]This work has been partially supported by the Spanish Science and Technology Commission (CICYT) under contract TIC96-0721-C02-01 (project "Analysis of spatial constraints and its application to mechanical design and robotic task simulation").

ficient computation of the separating plane with time coherence;

- and the *applicability constraints* [7] which permit detecting those vertex-face and edge-edge pairings that can really come into contact, if only translational motions are permitted.

These strategies rely on different criteria, some of which may be combined. Note, however, that the two latter methods work strictly for convex polyhedra. In particular, the last one has been exploited in [12], where a geometric representation of the applicability relationships, called Spherical Face Orientation Graph (SFOG), was developed to efficiently restrict the search to those edge-face pairs that may contact first, when the objects are initially disjoint and their relative orientation is known. Experimental results showed a linear growing of the number of edge-face candidates with the complexity of the involved polyhedra (instead of the quadratic number of all possible edge-face combinations).

The present work deals with the extension of this strategy to nonconvex polyhedra. The particular features of the SFOG representation in the nonconvex case require an algorithmic treatment different from the one in the convex case. The proposed new algorithms, together with the experimental results that we have obtained, constitute the main contributions of this work.

The paper is structured as follows. For self-containment, Sections 2 and 3 review the items on which the present work is based: the edge-face intersection test, that allows a decomposition-free interference detection between nonconvex polyhedra, and its associated lower distance bound function [15], as well as the geometric representation of the applicability constraints [12] that constitutes the input to the algorithms explained in Section 4. These algorithms are based on the plane sweep paradigm. Experimental results are displayed in Section 5, concerning not only the savings in computational effort provided by the proposed strategy, but also the improvement in the quality of the lower distance bound function. Conclusions are given at the end of the paper, as well as further lines of research.

2 Decomposition-free interference detection and distance computation

The typical edge-face intersection test contemplates only the case of the face being a convex polygon [4]. This means that a nonconvex face has to be split into convex parts, and the test has to be applied between the edge and each one of these parts. This decomposition can be avoided if the edge-face intersection predicate in [15] is used, as described in what follows.

For edge e to intersect a possibly nonconvex face f, two conditions must simultaneously hold:

- both endpoints of the edge, $\partial^+ e$ and $\partial^- e$, must be in opposite halfspaces, of those defined by the plane that contains the face f, and

- the line supporting the edge e must intersect the face f.

This test can be realized through a logical combination of the truth values of *basic contact predicates.* Any contact between two polyhedra can be described in terms of the two basic contacts: the vertex - face and the edge - edge contact. The *basic contact functions* $A_{v,f}$ and B_{e_1,e_2} are computed using the coordinates of the features involved in these basic contacts, and the *basic contact predicates* $\mathbf{A}_{v,f}$ and $\mathbf{B}_{e_1,e_2}$ are the truth values associated to the signs of the basic contact functions.

The two parts of the intersection predicate can be easily identified (see [15] for details):

$$(\mathbf{A}_{\partial^+ e,f} \oplus \mathbf{A}_{\partial^- e,f}) \wedge$$
$$[\bigoplus_{e_f \in \partial f} (\mathbf{A}_{\partial^+ e_f,f_e} \oplus \mathbf{A}_{\partial^- e_f,f_e}) \wedge (\mathbf{A}_{\partial^- e_f,f_e} \oplus \mathbf{B}_{e,e_f})]$$

where $\oplus$ is the exclusive OR connector, and ∂f represents the set of edges in the boundary of face f.

To perform interference detection between the boundaries of two polyhedra, this predicate has to be applied to all possible edge - face pairings.

The values of the basic contact functions $A_{v,f}$ and B_{e_1,e_2} correspond to primitive vertex - plane and line - line distances, respectively. If they are combined in the same way as in the predicate above, using functional operators instead of logical ones, a lower bound on the distance between the edge and the face is obtained:

$$\begin{aligned}&min(smin(A_{\partial_e^+,f}, A_{\partial_e^-,f}),\\ &smin_{e_f \in \partial f}(min(smin(A_{\partial_{e_f}^+,f_e}, A_{\partial_{e_f}^-,f_e}),\\ &\qquad\qquad smin(A_{\partial_{e_f}^-,f_e}, B_{e,e_f}))))\end{aligned}$$

where *smin* is a parity function corresponding to the exclusive OR connector.

If the edge and the face do not intersect, the sign of this function is negative. A lower bound on the distance between the polyhedra is computed by taking the maximum value of all these functions (for all the edge - face combinations). This has been stated in [15] and has been formally proved in [10], where the degenerate case in which the function reports contact (is equal to zero) while the polyhedra may actually be apart, has been analyzed.

3 Geometric representation of applicability constraints

The *applicability constraints* were developed in [7] as necessary and sufficient conditions for the two basic contacts (vertex - face and edge - edge) between the features of two convex polyhedra whose relative orientation does not change. These constraints allow to restrict considerably the set of edge - face candidates for first intersection: if the contact of a vertex and a face is applicable, only one of the adjacent edges to the vertex has to be considered for intersection with the face, whereas for an applicable edge - edge contact, the candidates are the edges themselves and their adjacent faces. Despite specific worst cases, where the number of candidates generated by the applicability constraints is quadratic (in the number of features of the involved polyhedra), experimental evidence exists for the reduction of the set of candidates to a linear size in the convex case [12].

As nonconvex polyhedra are considered, the applicability constraints become necessary but not sufficient conditions for contact. A contact between features for which the applicability conditions hold is now said to be *locally applicable*, as other features of the polyhedra –not considered in the applicability conditions– may prevent the contact from being actually realizable. False candidates for edge - face intersection may arise, thus leading towards a conservative strategy.

In order to obtain efficiently all the applicable vertex - face and edge - edge pairings, a suitable representation needs to be used. In [12], we developed the **Spherical Face Orientation Graph (SFOG)**: a dual representation of a polyhedron on the unit sphere of orientations, with an explicit depiction of the adjacency relationships. A node on the sphere stands for a face (as the direction of the outgoing vector of the plane that supports this face), an arc on a maximal circle joining two nodes represents the edge shared by the corresponding faces (minor arcs correspond to convex edges, major arcs to concave ones), and a vertex is represented by a cycle of arcs and nodes (i.e., the adjacent edges and faces) which bound a region on the sphere if the vertex is convex or pseudo-convex (called *convex subregion* in the latter case). These ideas are depicted in the leftmost column of Figure 1, where the SFOGs of a tetrahedron (top) and of a rectangular prism (bottom) are displayed, and in Figure 2 which illustrates the particularities attached to nonconvex polyhedra: concave arcs, convex subregions (*csr*) corresponding to local convex hulls, and possible overlapping of convex subregions.

The applicability relationships can be determined by superimposing the SFOG of one polyhedron on the central symmetric image of the other one (Figure 1), and determining all the node-in-region inclusions (that stand for the corresponding applicable vertex - face pairs), and the convex arcs intersections (applicable edge pairs). Once the applicable feature pairs have been found, determining the edge - face candidates for first intersection is straightforward, as mentioned above.

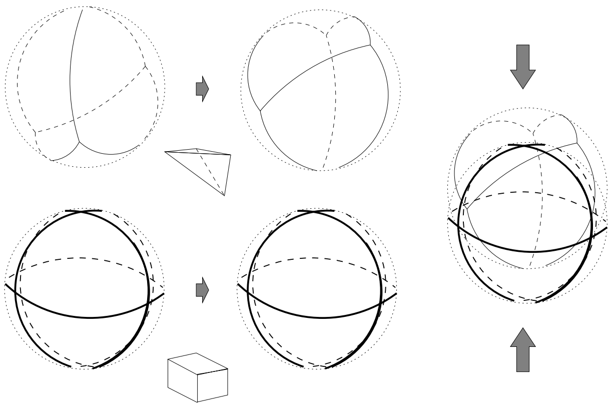

Figure 1: *Overlay of the SFOGs corresponding to two polyhedra in order to obtain a compact representation that allows to determine the applicability relationships. The SFOG of a rectangular prism (heavy lines) is superimposed on the central symmetric image of the SFOG of a tetrahedron (fine lines).*

The issue is now to come up with an efficient algorithm to determine these node-in-region inclusions and intersections between arcs. In the convex case, this can be done by exploiting the connectivity of the representation, as described in [12].

Connectivity, however, cannot be exploited in the nonconvex case: concave arcs do not permit a progressive exploration of the sphere, and they give no information concerning applicability. In other words, concave arcs are better eliminated. This leads towards a setting where we have two sets of possibly disconnected arcs and nodes, where all the intersections between the arcs of the two sets have to be determined (as well as the node-in-region inclusions), and where intersections between arcs belonging to the same set may exist. This can be solved by means of a spherical sweep, as explained in the next section.

4 Algorithms for finding applicable edge-face pairs

Algorithms to determine line segment intersections in the plane have been extensively studied in the field of Computational Geometry, all relying on the plane sweep technique. Therefore, we

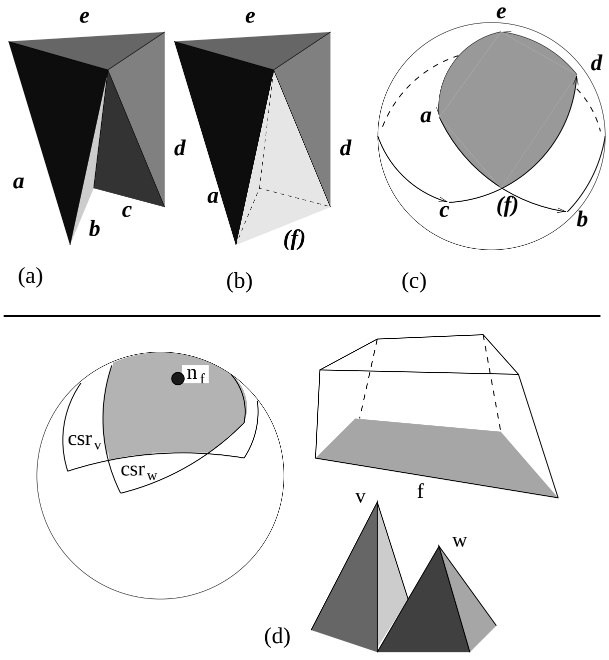

Figure 2: *(a) A pseudo-convex vertex, (b) its local convex hull, and (c) the corresponding* csr. *(d) Overlapping convex subregions, and a face (that belongs to the other polyhedron) which is applicable to the two vertices.*

just have to adapt those algorithms to sweep a spherical surface.

4.1 Brief review of plane sweep algorithms

The plane sweep technique (also known as line sweep or sweep line technique), can be described as follows: a sweep-line, assumed w.l.o.g. to be vertical, is swept through the whole plane. At a given instant, the sweep-line is intersecting some segments of the considered set. All the segment intersections to the left of the sweep-line have already been computed and will not be affected by subsequent intersections to the right. The sweep-line introduces in a natural way an adjacency relationship between the segments it intersects. This allows to consider for intersection only the adjacent segment pairs (contrarily to the brute-force approach which considers all possible pairings). These adjacency relationships change each time a segment endpoint is encountered or two segments intersect. Therefore, the continuous sweep process can be discretized by means of the *event point schedule*, i.e., the sequence of abscissae that correspond to the segments' endpoints as well as the intersection points, and the structure that maintains the *sweep line status* will allow for queries concerning adjacency relationships, i.e., segment intersection candidates.

The problem we are facing consists in determining the intersections between arcs of two sets (say "red" and "blue" arcs), where arcs inside the same set may also intersect, although we are not interested in determining these $k_{r-r} + k_{b-b}$ "monochromatic" intersections. This restricts considerably the type of algorithms to be considered in the whole taxonomy of segment intersection detection procedures (see [10] for a description of this taxonomy and related references).

In [2] an algorithm is described that reports all k_{r-b} red - blue (or "purple") intersections between line segments in time $O((n_r\sqrt{n_b} + n_b\sqrt{n_r} + k_{r-b})\log(n_r + n_b))$. This complexity is improved in [1] to overall $O(n^{4/3}\log^{(\omega+2)/3} n + k_{r-b})$ time and using $O(n^{4/3}/\log^{(2\omega+1)/3} n)$ space, where ω is a constant < 3.33 and $n = n_r + n_b$, by applying a divide-and-conquer strategy. This algorithm is deterministic; using random-sampling techniques, an expected time of $O(n^{4/3}\log n + k_{r-b})$ is obtained [1].

Using recently developed data structures that allow to maintain a partial order on the line segments, an expected time of $O((n + k_{r-b})\alpha(n)\log^3 n)$ can be obtained in the case where the sets of red and blue segments are connected [3]. If red and blue segments are grouped into c_r and c_b connected components, the time becomes $O((c_b n_r + c_r n_b + k_{r-b})\log^3 n\alpha(n))$. This algorithm is not as hard to implement as the previous one, and we have adapted it to sweep the sphere (see [10] for details about the algorithm and its implementation). Nonetheless, the use and maintenance of the involved sophisticated data structures is highly time-consuming, which renders this algorithm less efficient than a naïve strategy (described in the following section) when applied to objects of moderate complexities as those used in our testbed. It is worth mentioning, though, that the savings in arc intersection tests increases faster than the burden of data structure management, implying that there exists a threshold scene complexity beyond which the sophisticated algorithm would always outperform the naïve one [10].

4.2 Naïve spherical sweep algorithm

It is straightforward to adapt the plane sweep principle to the sphere: the vertical sweep-line is replaced by a sweep-meridian, the sweep begins at an arbitrary point (as we cannot speak of a "leftmost" point), and proceeds eastwards (as the plane sweep from left to right). As in the planar case of line segment intersection detection, two arcs will intersect at most at one point, and monotonicity is ensured: one arc cannot intersect the sweep-meridian at more than one point simultaneously.

Each time the sweep-meridian arrives at the western endpoint of an arc $a[i]$, this arc is tested for intersection with all the active arcs (i.e., arcs currently intersected by the sweep-meridian) of opposite color $L_{\overline{c}[i]}$, and included in the list of active arcs $L_{c[i]}$ ($c[i]$ denotes the color of $a[i]$). As soon as the eastern endpoint of an arc is reached, that arc is deleted from the active list. In this way, every purple intersection will be detected exactly once. A pseudo-code transcription of this simple algorithm is presented:

Naïve SFOG search algorithm

```
Preprocessing: Order the 2n endpoints e[i] by increasing meridian value. Let a[i] be the arc with endpoint e[i] and c[i] its color.
for ($i = 1..2n$) do
    if (e[$i$] is a western endpoint) then
        for (all $a' \in L_{\overline{c}[i]}$) do
            if ($a[i] \cap a'$) then
                report purple intersection
            endif
        endfor
        insert($a[i], L_{c[i]}$)
    else
        delete($a[i], L_{c[i]}$)
    endif
endfor
```

As said before, the "first" endpoint is an arbitrary choice and, at this first instant, no lists of active arcs exist. Therefore, a second sweep will have to be performed to take into account all the purple intersections with arcs that are still active after the last endpoint. Figure 3 depicts the purple intersections that can be detected along the first sweep and those which cannot be determined if no second sweep is performed.

As for node-in-region inclusions, they are computed during the same sweeping operation: each region defines an interval on the sweep-line and it has to be determined which intervals of the opposite color include the current endpoint. Each interval is defined by the upper and lower arcs bounding the region at every instant. Regions begin and end at given (not necessarily all) endpoints. The regions a, say, red node belongs to are computed by determining the blue arcs that cut the sweep-line above this node and then finding out if the regions underneath these arcs are also bounded by arcs that cut the sweep-line below that node.

5 Experimental results

Experiments have been carried out to quantify the benefits derived from the use of applicability constraints. Algorithm A implements the testing of all possible edge-face pairings for intersection

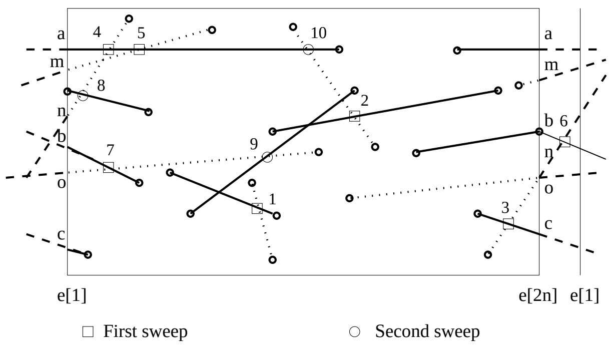

Figure 3: *Plane analogue of the spherical sweep. Red and blue arcs are depicted as dotted and solid segments. Some of them (a, b, c, m, n, o) begin before or at the last event point n_t and end after the first one n_1. These segments are activated during the first sweep from e[1] to e[2n], where some purple intersections can already be detected (marked with a small square), and their intersections with the segments that had ended before they where generated (marked with an empty circle) can only be detected during a second sweep. Numbers indicate the order in which the intersections are computed.*

(the pairs of polyhedra are supposed to be disjoint), and algorithm B is the implementation of the naïve sphere sweep algorithm, plus the interference tests on the edge - face pairs obtained.

Algorithms A and B have been tested on a set of nonconvex polyhedra ranging from a pentahedron (with only one concave edge) to an hour-glass shaped polyhedron (160 edges, including 32 concave ones). The results are displayed below. Figure 4 shows a drastical reduction in the number of edge - face intersection tests to perform for the considered objects. Figure 5 displays the execution times (on a SG O2 workstation, SPEC int95 4.8, SPEC fp95 5.4) of algorithms A and B, as well as the time needed by the preprocessing step of B. We have to insist that this preprocessing has to be done only once. Note also that in both figures a logarithmic scale is used.

The benefits of applicability constraints are not restricted to the lowering of the computational effort of interference detection, but have also a positive effect on the computation of a lower bound on the distance between the polyhedra. It is not difficult to prove that applicability constraints do not eliminate all edge - face pairs that can provide lower distance bounds, as has been done in [10].

Besides the savings in the computational effort needed to compute this lower distance bound, which mimics those for the interference detection case, experimental results show an improvement in the quality of the distance bound (Figure 6), in the sense that this value, computed considering only the candidate edge - face pairs, is in general closer to the real distance between the polyhedra than the lower distance bound computed considering all the edge - face pairs (in the worst case, both values are equal).

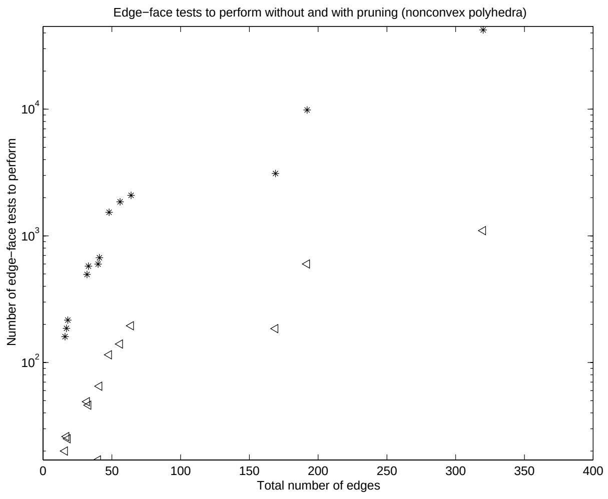

Figure 4: *Comparison between the total number of elementary edge - face tests to perform without applicability pruning, i.e. algorithm A (*), and with a previous pruning step, algorithm B (◁), in settings that include only nonconvex polyhedra.*

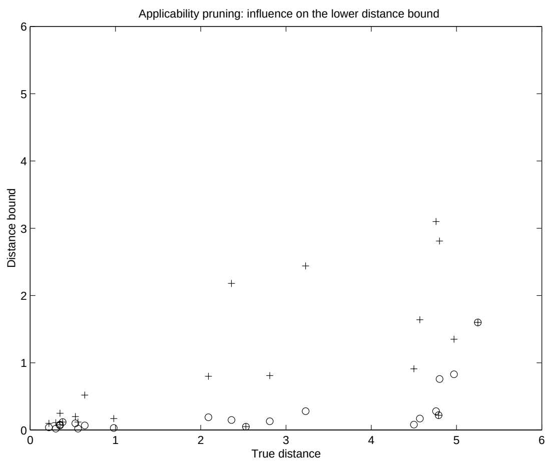

Figure 6: *Comparison of the lower distance bound vs. the real distance between the solids. Although the quality of the distance bound depends mainly on the geometry of the objects, it is also clear that, in general, the closer the objects are, the better the quality of the distance bound obtained. In some cases, the lower distance bounds obtained with (+) and without (o) applicability pruning coincide, but the general tendency is of a much better performance in the former case.*

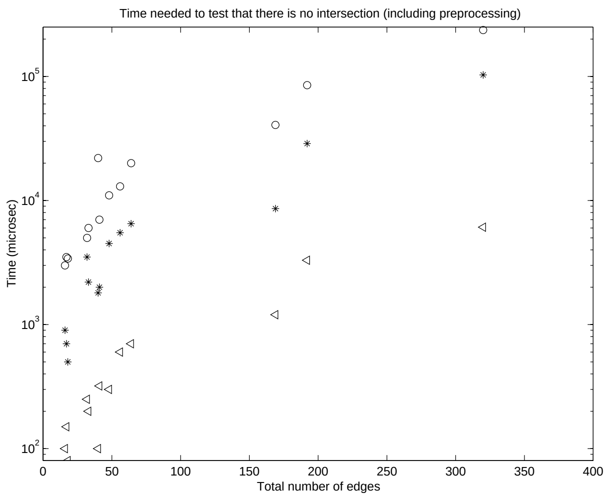

Figure 5: *Comparison of the execution times of algorithm A (*), and algorithm B (◁), with a previous pruning step (o), in settings that include only nonconvex polyhedra.*

6 Conclusions and further research

Nonconvex polyhedra can be tested for interference in an efficient way, without decomposing them into convex parts. Applicability pruning is a preprocessing step, as it is the traditional solution of the convex decomposition of the original polyhedra, but it does not increase the complexity of the setting, as no fictitious features are introduced. Experimental evidence has been obtained for the increment in computational efficiency derived from this pruning: interference detection based on edge - face intersection tests performs 10 to 100 times faster if a previous selection of candidate pairs based on the applicability relationships is performed. Benefits extend to the related issue of computing a lower bound on the distance between the polyhedra, whose quality is improved.

The next step is the integration of the static interference method in the dynamic domain, for collision detection. In [11] details were given about such an integration: the trajectory parameterization approach to collision detection was formulated in terms of the parameterized basic contact functions and their combination in the edge - face intersection predicate. This method, as well as the conceptually simpler approach of multiple interference detection, benefit from the savings in computational effort provided by the applicability pruning step. But an important question has to be addressed now: the applicability relationships are orientation-dependent, which means that along trajectories that entail a change in the relative orientation of the polyhedra, new edge - face candidates arise, while others become no longer valid. Therefore, one must be able to determine the *intervals of isoapplicability*, i.e., the ranges of relative orientations –along the trajectory– for which the same applicability relationships hold. In the trajectory parameterization approach, this means to determine the values of the parameter where these changes occur, whereas in the multiple interference detection approach a discretization of time based on isoapplicability will have to be considered, besides the standard discretization based on distance and relative velocities. The SFOG representation can be used to this end: the intervals of isoapplicability are delimited by the *rotation events*, each time a node of one SFOG crosses an arc of the other one. These questions are addressed in [10], but devising an efficient method for computing all the rotation events for arbitrary changes in the relative orientation of the polyhedra is still an open issue.

References

[1] P. Agarwal. Partitioning arrangements of lines. ii: Applications. *Discrete Comput. Geom.*, 5:533–573, 1990.

[2] P. Agarwal and M. Sharir. Red-blue intersection detection algorithms, with applications to motion planning and collision detection. *Siam J. Comput.*, 19(2):297–321, Apr. 1990.

[3] J. Basch, L. J. Guibas, and G. D. Ramkumar. Reporting red-blue intersections between connected sets of line segments. In *4th European Symposium on Algorithms*, pages 302–319, 1996.

[4] J. F. Canny. Collision detection for moving polyhedra. *IEEE Trans. Patt. Anal. Mach. Intell.*, 8(2):200–209, Mar. 1986.

[5] K. Chung and W. Wang. Quick collision detection of polytopes in virtual environments. In *Proceedings of ACM Symp. on Virtual Reality Software and Technology*, pages 1–4, July 1996. http://www.cs.hku.hk /˜tlchung /collision_library.html.

[6] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. In *Proceedings of ACM Int. 3D Graphics Conference*, volume 1, pages 189–196, 1995. http://www.cs.unc.edu /˜geom /I_COLLIDE.html.

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

[8] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96*, 1996. http://www.cs.unc.edu /˜geom/OBB /OBBT.html.

[9] T. C. Hudson, M. C. Lin, J. D. Cohen, S. Gottschalk, and D. Manocha. V-collide: Accelerated collision detection for vrml. In *Proceedings of VRML*, 1997. http://www.cs.unc.edu/˜geom/V_COLLIDE .html.

[10] P. Jiménez. *Static and dynamic interference detection between non-convex polyhedra*. PhD thesis, Universitat Politécnica de Catalunya, 1998. http://www-iri.upc.es/people/jimenez/phdthesis.html.

[11] P. Jiménez and C. Torras. Collision detection: a geometric approach. In *Modelling and Planning for Sensor Based Intelligent Robot Systems*, pages 68–85. World Scientific Pub. Co., Nov. 1995.

[12] P. Jiménez and C. Torras. Speeding up interference detection between polyhedra. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1485–1492, Minneapolis (MN), Apr. 1996.

[13] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1008–1014, Sacramento (CA), 1991.

[14] M. K. Ponamgi, D. Manocha, and M. C. Lin. Incremental algorithms for collision detection between polygonal models. *IEEE Trans. on Visualization and Comp. Graphics*, 3(1):51–64, 1997.

[15] F. Thomas and C. Torras. Interference detection between non-convex polyhedra revisited with a practical aim. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 587–594, San Diego (CA), May 1994.

[16] G. Vanecek. Back-face culling applied to collision detection of polyhedra. *Journal of Visualization and Computer Animation*, 5(1):55–63, January-March 1994.