Distance Computation between Non-convex Polyhedra at Short Range Based on Discrete Voronoi Regions

Katsuaki Kawachi and Hiromasa Suzuki
Department of Precision Machinery Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{kawachi, suzuki}@cim.pe.u-tokyo.ac.jp

Abstract

An algorithm for calculating the minimum distance between non-convex polyhedra is described. A polyhedron is represented by a set of triangles. In calculating the distance between two polyhedra, it is important to search efficiently the pair of the triangles which gives the pair of closest points.

In our algorithm discrete Voronoi regions are prepared as voxels around a non-convex polyhedron. Each voxel is given the list of triangles which have possibility to be the closest to the points in the voxel. When a triangle on the other object is intersecting a voxel, the closest triangles can be efficiently searched from this list on the voxel.

The algorithm has been implemented, and the results of distance computations show that it can calculate the minimum distance between non-convex polyhedra composed of a thousand of triangles at interactive rates.

1 Introduction

Computing distance between objects is an important problem in various field including computer graphics animation and simulation. In order to prevent the moving objects from interference, it is needed to detect if the objects are colliding or not. In addition to this interference detection, it is useful to calculate the minimum distance between the objects for predicting the instance of collision in dynamics simulation.

The algorithms of distance computation can be classified by their targets as follows.

- Non-polyhedral Objects
  - CSG models
  - Curved surfaces
  - Clouds of points
- Polyhedral Objects *without* topological information
  - Sets of Polygons
- Polyhedral Objects *with* topological information
  - Convex polyhedra
  - Non-convex polyhedra

In this paper we present an algorithm for polyhedral objects. The object consists of a set of triangles and is not necessarily needed to have topological information.

2 Related Work

For distance computation between convex polyhedral models specified by a boundary representation, there are two major algorithms: closest-feature algorithms [7][8] and simplex-based algorithms [2][1].

The Lin-Canny closest feature algorithm [7] defines Voronoi region on its features (faces, edges, and vertices). If two features are closest, each feature is inside the Voronoi region of the other. In dynamics simulation the algorithm can efficiently find the closest features by tracking the pair of the closest features. V-Clip [8] employs the improved closest feature algorithm which can handle the penetration between objects and calculate the distance robustly. In these closest feature algorithms the method of searching the closest feature pair utilizes Voronoi regions defined around convex polyhedra. Unfortunately these algorithms are based on convex polyhedra and cannot be directly applied to non-convex ones.

In this paper, we deal with the non-convex problem. We utilize Voronoi regions for picking up the closest triangles between non-convex polyhedra. Because it is difficult to calculate accurate structure of the Voronoi regions on a non-convex polyhedron, they are defined by the discrete approximation method using voxels [6].

3 Discrete Voronoi Regions

In this section we describe the definition of the discrete Voronoi regions and the algorithm of distance computation between non-convex polyhedra.

3.1 Cost of Distance Computation

The total cost of interference detection between two polyhedra can be formulated as the following equation [3]:

$$T = N_v \times C_v + N_p \times C_p, \tag{1}$$

where

T : total cost function for interference detection
N_v : number of bounding volume pair overlap tests
C_v : cost of testing a pair of bounding volumes for overlap
N_p : number of feature pairs tested for interference
C_p : cost of testing a pair of features for interference

As C_p is expensive in general, bounding volume with smaller C_v can save T by decreasing the number N_p. Typical algorithms for defining bounding volumes include axis-aligned boxes, octrees [10], sphere trees [9], OBBTree [3], swept sphere volumes [5] etc. Avobe all, the PQP library utilizing swept sphere volumes can perform efficient distance calulation by transforming distance computation to intersection detection.

In calculating the minimum distance between non-convex polyhedra the simplest algorithm tests the all pairs of features in them and picks up the pair with the smallest distance. In the same way of collision detection, because the cost of distance computation between a pair of features is expensive, the total cost can be saved by reducing the number of pairs of features with some bounding volume.

3.2 Voxels around Objects

A Voronoi region can be defined for each feature on a polyhedron:

Definition A Voronoi region associated with a feature is a set of points closer to that feature than any other.

Figure 1 shows a two dimensional illustration of the Voronoi regions defined around a convex object. If a feature X on another polygon is included in region R_b (Figure 2), it is closest to the edge b, and it is needless to calculate its distance against the other features a, c. The regions are easily defined on a convex polyhedron, and I-Collide [7] and V-Clip [8] utilize the Voronoi regions for calculating the distance between convex polyhedra.

If we can define Voronoi regions on the features of a non-convex polyhedron, a feature X of the other polyhedron can

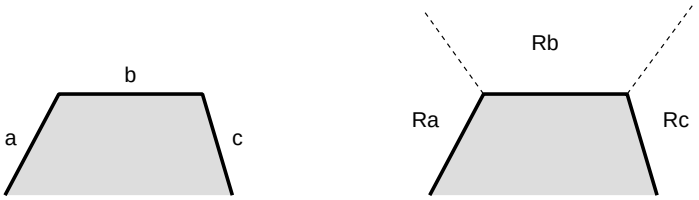


Figure 1. Voronoi Regions Defined around Convex Polygon

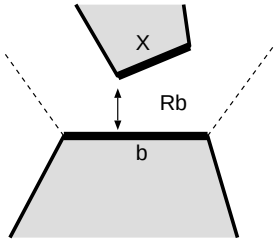


Figure 2. Closest Feature in Voronoi Regions

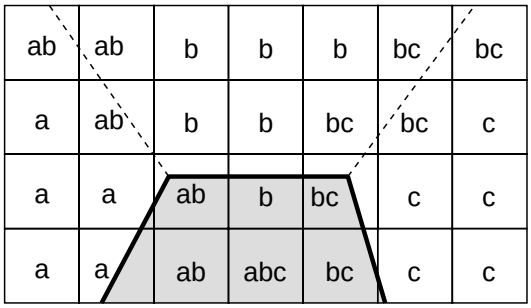


Figure 3. CFL (Closest Feature List) on Voxels

be closest to the feature Y whose Voronoi region includes X. Because it is difficult to calculate accurate structure of the Voronoi regions of a non-convex polyhedron, we discretely define Voronoi regions by voxelizing the space around the polyhedron and do not explicitly calculate the structure of Voronoi regions. Each voxel is given the list of the features which have possibility to be closest to the voxel (Figure 3). This list is called "Closest Feature List" (CFL). If the feature X has interference with the voxel V, we need to calculate the distances between X and the features in CFL on the voxel V.

3.3 Representation of Non-convex Polyhedra

In this paper we present an algorithm for calculating distance between objects with the following properties:

- A polyhedron is represented by a set of triangles. In the closest feature algorithms, a polyhedron is constituted by three features (faces, edges, and vertices). In our algorithm edges and vertices are not regarded as independent features but as a part of a triangle. By unifying them to triangles, we can decrease the number of distance calculations.
- A triangle does not necessarily have topological information about neighboring triangles across the edges. If topology is known, it is utilized for culling the voxels which cannot include the triangle on their CFLs.
- Triangles in a polyhedron may have interference each other.

These properties are useful for dealing with polyhedra with boundaries, messy polygon models etc. As the algorithm does not calculate the Voronoi regions explicitly, triangles in a polyhedron can be freely arranged. The algorithm does not distinguish the front and back of a triangle. So the distance calculated by the algorithm is always zero or more than zero.

3.4 Closest Triangles to Voxel

In creating CFL on voxels, we utilize the minimum and maximum values of the distance between a point in a voxel and a triangle.

If a point x and triangles are given, one triangle which is closest to the point x can be chosen by the relative position of x to the triangles. If x exists inside a voxel V, the closest triangle varies as x moves in V. CFL on a voxel can be created by picking up all the closest triangles for the all points in voxel V. If the triangle Y on the other polyhedron is interfering with V, the triangles in the list on V have possibility to be closest to Y.

Given a point x and a triangle A, we can calculate their minimum distance $d_A(x)$. In voxel V there exists the points $x_A^{\min}$ and $x_A^{\max}$ which give the minimum and maximum value of $d_A(x)$ (Figure 4). By this definition, for all points x in the voxel V we can write

$$d_A(x_A^{\min}) \leq d_A(x) \leq d_A(x_A^{\max}),\ \forall x \in V. \qquad (2)$$

In the same way, the minimum distance $d_B(x)$ can be defined for another triangle B, and we get the points $x_B^{\min}$ and $x_B^{\max}$ satisfying the following condition:

$$d_B(x_B^{\min}) \leq d_B(x) \leq d_B(x_B^{\max}),\ \forall x \in V. \qquad (3)$$

If the points $x_A^{\max}$ and $x_B^{\min}$ satisfies $d_A(x_A^{\max}) < d_B(x_B^{\min})$, the definitions (2) and (3) leads:

$$d_A(x) < d_B(y),\ \forall x \in V,\ \forall y \in V, \qquad (4)$$

that is, any points in V are given smaller minimum distance by the triangle A than B. Consequently the triangle B cannot be closest to any points in V.

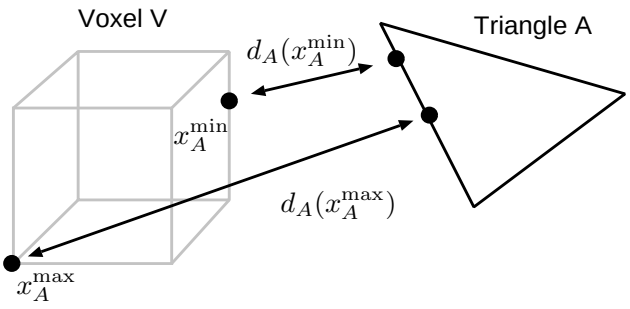


Figure 4. Minimum Distance between Triangle and Point in Voxel

In picking up all the closest triangles for the voxel V, the minimum value $d_i^{\min}$ and maximum value $d_i^{\max}$ of the distance d_i between V and the triangle i are calculated for all triangles at first. Then the triangles are sorted by their minimum values of distance. If some triangles have the same minimum value, they are sorted again by the maximum values of distance. By this procedure we can get the triangle k with the smallest $d_k^{\min}$ and $d_k^{\max}$. By the condition (4), the triangle i satisfying $d_k^{\max} < d_i^{\min}$ cannot be closest to the voxel V and is not included in CFL on V.

In our algorithm voxels are defined on the local coordinate system of a non-convex polyhedron. The CFLs are calculated only once as a preprocess of distance computations. They are kept with the geometric information of the polyhedron and are reused when the relative orientation of the objects is changed, because the CFL calculation generally takes much time. For instance, the CFLs on $30 \times 30 \times 30$ voxels around a polyhedron with 1314 triangles requires about 40 minutes to compute. The CFL calculation can be much accelerated if we utilize the computation method of Voronoi region using graphics hardware [4].

3.5 Calculating Distance with CFL

With CFLs we compute the distance between non-convex polyhedra A and B with the following four steps.

At the first step, the voxels on A which have interference with the triangles in B are picked. The interference test is done efficiently by defining a BSP tree of voxels on A and an OBBTree [3] on the triangles of B.

At the second step, the closest feature tests are applied for the intersecting triangles and voxels found in the first step. The Lin-Canny closest feature algorithm utilizes the necessary and sufficient condition that each feature is inside the

Voronoi region of the other. This condition can be rewritten for the discrete Voronoi regions with CFLs as follows:

$$\left\{\begin{array}{l} \text{Feature } X \text{ intersects voxel } V_Y \\ \text{(the CFL of } V_Y \text{ includes feature } Y\text{)} \\ \quad \textit{and} \\ \text{Feature } Y \text{ intersects voxel } V_X \\ \text{(the CFL of } V_X \text{ includes feature } X\text{)} \end{array}\right. \quad (5)$$

Because the condition(5) is not sufficient but necessary for non-convex polyhedra, if a voxel V on A intersecting a triangle T in B are found at the first step, they can not be closest and are ignored at the second step when the condition(5) is not satisfied.

In order to check the condition, the voxels on B which have interference with the triangles in A are picked at the second step. If a triangle in the CFL of V has interference with the voxel with the CFL including T are found, the condition is satisfied.

At the third step, the picked voxels are sorted by their minimum values of distance $d_k^{\min}$, and at the final step, for each voxel in the order of $d_k^{\min}$, the minimum distance between the triangles in the CFL and the triangle interfering the voxel is calculated.

By sorting the voxels at the third step, we can efficiently eliminate the excess voxels. If in the way of final step we have a voxel k with distance $d_k^{\min}$ which is greater than the minimum distance given by the previous voxels, we can cut off the distance computations for the following voxels, which cannot be the closest to the polyhedron any more.

In the implementation of our algorithm, voxels are defined in the region which is twice as large as an object's bounding box. If the object is out of this region, the distance is approximated by that of the convex hull of the original polyhedra.

3.6 Voxel Culling

If topological informations between triangles are available, the voxels where the triangle can be closest can be culled by the approximated Voronoi region on the triangle. The voxels which have no interference with the region cannot be closest and do not need to be calculated their distances to the triangle.

The Voronoi region of the triangle is approximately defined by the planes containing the edges on the triangle. Each plane is parallel to the average of the normal vector of the triangle and that of the neighboring triangle.

When the polyhedron is convex, the approximated region on a triangle is identical to the accurate Voronoi region. When non-convex, it is not necessarily identical, but it includes the accurate Voronoi region, so the approximation by neighboring triangles does not make an incomplete CFL.

As shown in Figure 5, the approximated Voronoi region for a triangle is partitioned with planes on edges (plotted with broken lines). The voxels interfering with the region can be the closest to the triangle.

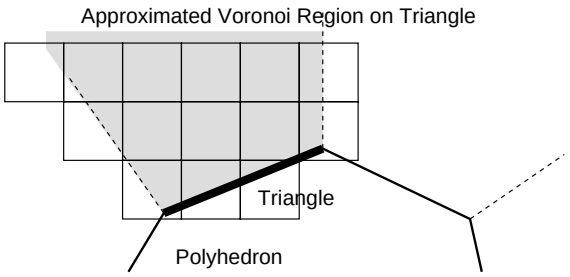


Figure 5. Voxel Culled by Approximated Voronoi Region

4 Results

Based on the algorithm of the discrete Voronoi regions we have implemented a program for calculating the minimum distance between non-convex polyhedra.

4.1 Length of CFL

Table 1 shows the number of voxels and the average length of CFL defined on a bunny model of 1314 triangles (Figure 6). The average length of CFL per voxel is affected by the size of the voxel. If the size of a voxel is increased, more triangles per voxel are listed. It can be said that smaller voxels make the length of CFL shorter and that the efficiency of distance calculation can be improved, but we have found that the length of CFL do not decreases very rapidly when the size of voxels is smaller than the size of triangles. The change of triangles picked for distance calculation in some voxel sizes is shown in Figure 7 as dark triangles. The thick line between two objects indicates the pair of closest points.

CFL on voxels are calculated on each polyhedron as a preprocess of distance calculation. For instance, a dark triangle on the back of the bunny is given in CFL on the black voxels (Figure 8).

Table 1. Number of Voxels and Length of CFL

Number of Voxels	Average Length of CFL
$10 \times 10 \times 10$	29.08
$15 \times 15 \times 15$	11.92
$20 \times 20 \times 20$	10.90
$30 \times 30 \times 30$	6.35

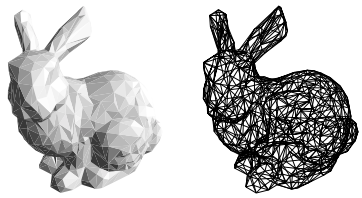
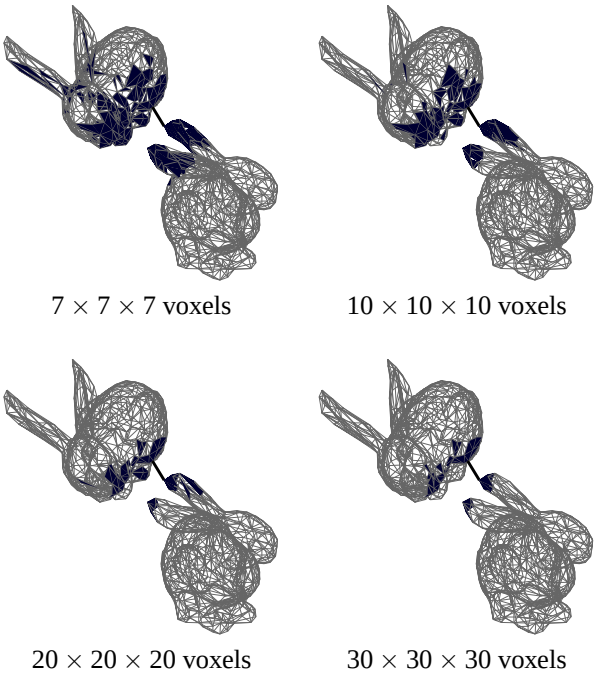


Figure 6. Non-convex Polyhedron with 1314 Triangles



$7 \times 7 \times 7$ voxels $10 \times 10 \times 10$ voxels

$20 \times 20 \times 20$ voxels $30 \times 30 \times 30$ voxels

Figure 7. Picked Triangles and Size of Voxels

4.2 Distance Calculation with CFL

In calculating the distance with $30 \times 30 \times 30$ voxels, the total number of distance calculations between a pair of triangles is diminished to 172 on average, which is about 1/10000 compared with the number of all combinations of triangles. The average of CPU time consumed in distance computation is 0.06 second with MIPS R10000 250MHz. The triangles which are picked from CFL on voxels and calculated their distance are shown in dark color (Figure 9). The thick line between two objects indicates the pair of closest points.

The algorithm enables us fast computation of the distance between non-convex polyhedra which consist of a thousand of triangles, and we utilize this algorithm for our interactive rigid body dynamics simulator named "pVRML", which can simulate the dynamics of physically-based models defined in 3D scene of VRML1.0.

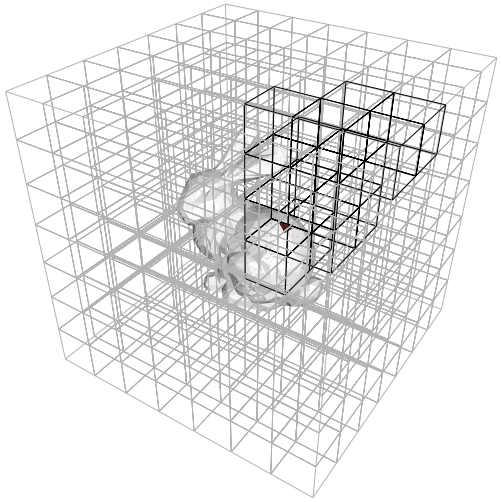


Figure 8. Voxels Associated with a Triangle

5 Conclusion and Future Work

We have described an the discrete Voronoi regions for calculating the distance between non-convex polyhedra. By defining the minimum and maximum values of distances between voxel and triangles, we can efficiently eliminate the voxels which cannot be closest. The algorithm enables us to calculate the distance between non-convex polyhedra made of a thousand of triangles at interactive rates, and we utilize it for an interactive dynamics simulator.

It is found that the average length of CFL is affected by the size of the voxel. In order to achieve more efficient distance computation algorithm, a method for deciding the optimal size of voxels for an polyhedron is needed.

If the polyhedra are interfering, our algorithm answers zero as the distance between them. In terms of dynamics simulation, it can be useful to extend the algorithm to treat penetration as minus distance in estimating the instance of collision.

Table 2 shows the comparison of the distance computation cost with the PQP library [5]. Our implementation is about 3.4 times as slow as PQP. The polyhedron in Figure 6 is used for this test, and CFLs are defined on $30 \times 30 \times 30$ voxels. This result is mainly caused by the large N_v, i.e. the number of intersection tests and closest feature tests between triangles and voxels performed in the first, second and third step described in the section 3.5, and these tests cost more than 70% of all computation time. By constructing a hierarchical structure of triangles in CFLs of voxels, they

can be more efficient and it is likely to be possible to reduce N_v as small as that of PQP .

Table 2. Performance of Distance Computation

Algorithm	Av. Computation Time	Av. N_p	Av. N_v
PQP	0.018 sec.	105.4	1781
CFL	0.061 sec.	171.6	7970

References

[1] S. Cameron. Enhanceing GJK: Computing minimum penetration distances between convex polyhedra. In *IEEE International Conference on Robotics and Automation*, April 1997.

[2] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, April 1988.

[3] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Computer Graphics Proceedings, Annual Conference Series*, pages 171–180. ACM SIGGRAPH, 1996.

[4] K. E. H. III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. In *Computer Graphics Proceedings, Annual Conference Series*, pages 277–285. ACM SIGGRAPH, 1999.

[5] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, Chapel Hill, 1999.

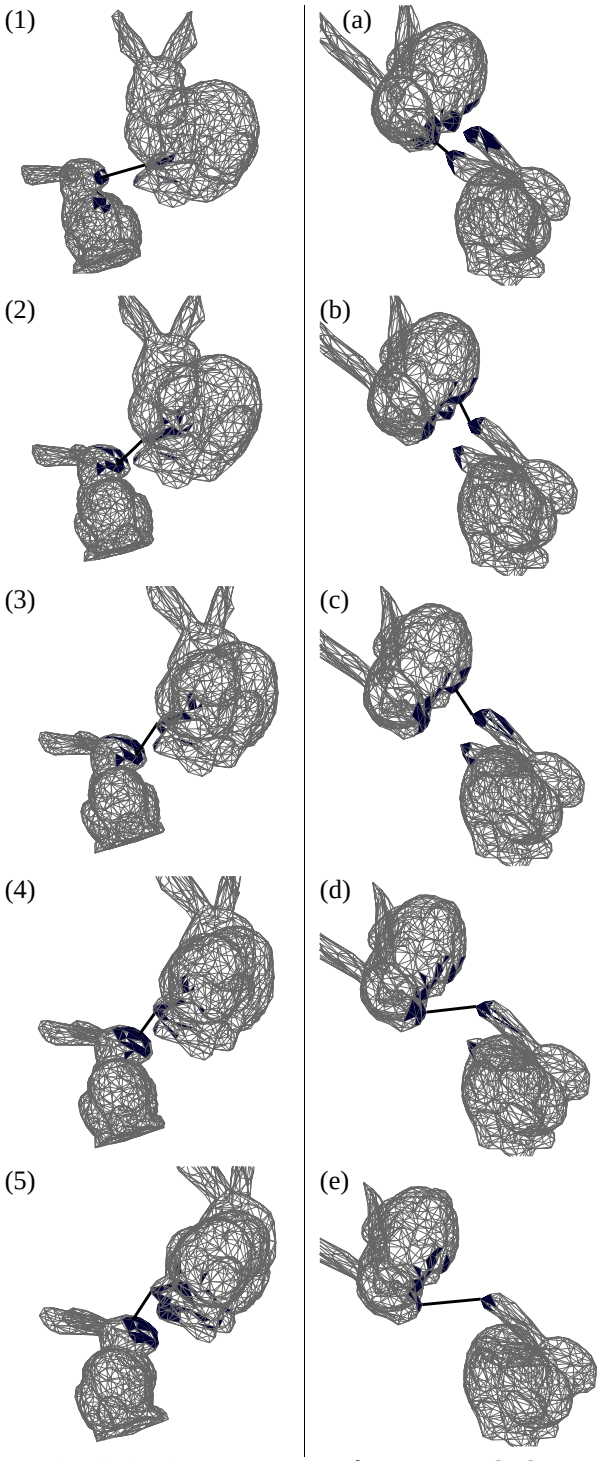
[6] D. Lavender, A. Bowyer, J. Davenport, A. Wallis, and J. Woodwark. Voronoi diagrams of set-theoretic solid models. *IEEE Computer Graphics and Applications*, 12(5):69–77, September 1992.

[7] M. C. Lin and J. F. Canny. Efficient algorithm for incremental distance computation. In *IEEE Conference on Robotics and Automation*, 1991.

[8] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR-97-05, MERL, July 1997.

[9] J. Pitt-Francis and R. Featherstone. Automatic generation of sphere hierarchies from cad data. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1998.

[10] H. Samet. *Spatial Data Structures: Quadtree, Octree and Other Hierarchial Methods*. Addison Wesley, 1989.



(1–5), (a–e): Sequences of Distance Calculation
Dark Triangles: Picked Triangles in Calculation
Thick Line: Pair of Closest Points

Figure 9. Closest Points between Two Moving Non-convex Polyhedra