

Fast and Robust Retrieval of Minkowski Sums of Rotating Convex Polyhedra in 3-Space*

Naama Mayer
Tel Aviv University
naamamay@post.tau.ac.il

Efi Fogel
Tel Aviv University
efif@post.tau.ac.il

Dan Halperin
Tel Aviv University
danha@post.tau.ac.il

ABSTRACT

We present a novel method for fast retrieval of exact Minkowski sums of pairs of convex polytopes in \mathbb{R}^3 , where one of the polytopes frequently rotates. The algorithm is based on pre-computing a so-called *criticality map*, which records the changes in the underlying graph-structure of the Minkowski sum, while one of the polytopes rotates. We give tight combinatorial bounds on the complexity of the criticality map when the rotating polytope rotates about one, two, or three axes. The criticality map can be rather large already for rotations about one axis, even for summand polytopes with a moderate number of vertices each. We therefore focus on the restricted case of rotations about a single, though arbitrary, axis.

Our work targets applications that require exact collision-detection such as motion planning with narrow corridors and assembly maintenance where high accuracy is required. Our implementation handles all degeneracies and produces exact results. It efficiently handles the algebra of exact rotations about an arbitrary axis in \mathbb{R}^3 , and it well balances between preprocessing time and space on the one hand, and query time on the other.

We use CGAL arrangements and in particular the support for spherical Gaussian-maps to efficiently compute the exact Minkowski sum of two polytopes. We conducted several experiments to verify the correctness of the algorithm and its implementation, and to compare its efficiency with an alternative (static) exact method. The results are reported.

1. INTRODUCTION AND RELATED WORK

Let P and Q be two polyhedra in \mathbb{R}^d . The Minkowski sum of P and Q is defined as $P \oplus Q = \{p + q \mid p \in P, q \in Q\}$. In this work we only deal with bounded *convex* polyhedra, which henceforth we refer to as polytopes, in \mathbb{R}^3 . Assume

*This work has been supported in part by the Israel Science Foundation (grant no. 236/06), by the German-Israeli Foundation (grant no. 969/07), and by the Hermann Minkowski-Minerva Center for Geometry at Tel Aviv University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPM 2010, Technion, Haifa, Israel

Copyright 2010 ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

that a polytope R is moving in three-dimensional space. It is well known that R will collide with another static polytope Q , if and only if the origin is in $P \oplus Q$, where P is a copy of R reflected about the origin. This observation is the basis of the intensive use of Minkowski sums in motion planning and many other related problems; see, e.g., [11].

Consider a motion planning application, where a robot moves amidst obstacles. The robot has to find a collision-free path to a goal position using translations *and* rotations. This problem has three degrees of freedom (dofs) in the plane and six dofs in three-dimensional space in the general case. When the robot is restricted to rotate about one axis only in three-dimensional space, the number of dofs reduces to four. We developed a data structure that can be used to efficiently and robustly retrieve the exact Minkowski sums that arise during this motion. This data structure is a major step toward efficient and robust collision detection during such motion.

It is well known that the Minkowski sum does not change under translation, while it can dramatically change under rotation. Hence, one way to deal with rotations is to recalculate the translational configuration-space of the robot to cope with rotation changes. Indeed, a common algorithm for computing the configuration space for a translating and rotating robot amidst obstacles is to compute the translational configuration-space for a sample of discrete robot-orientations [8, §13.5]. This is feasible in the plane, where the translational configuration-space can be represented by a two-dimensional arrangement. Computing the translational configuration-space in three-dimensional workspace is much more complicated. Applications tend to avoid computing the entire configuration-space. However, collisions still must be detected. A similar feasible approach for detecting collision between translating and rotating robots is to recompute the resulting sum from scratch for few discrete orientations along the robot rotation-orbit. The results of these procedures are only approximations, in the sense that they do not necessarily reveal all the *combinatorially distinct* structures (see below for a formal definition).

Our method considers in advance all the possible combinatorial structures of Minkowski sums under rotation about a given arbitrary axis, and creates a data structure, from which the appropriate Minkowski sum structure can be easily extracted for a given rotation-angle. Our implementation handles degenerate input and produces exact results. We only use exact number-types. We use algebraic number-types when rational numbers are not sufficient, though reducing the performance impact by using an efficient type of

exact algebraic numbers known as the sqrt-extension.

Minkowski sum is a common and practical operation with various applications in a wide variety of fields, such as computer graphics, motion planning, computer-aided-design, and computer-aided-manufacturing. Minkowski sums have been intensively investigated over the years; a large number of methods were proposed for efficiently constructing Minkowski sums. A few examples are the methods developed by Vardahan and Manocha [24], by Ghosh [13], by Kaul and Rossignac [19] and by Gritzmann and Sturfels [15]. Several output-sensitive methods were especially designed for exact construction of Minkowski sums of polytopes. Fogel and Halperin [11] represented every polytope as a Gaussian map and constructed their Minkowski sum from their overlay. Hachenberger [17] suggested to calculate Minkowski sums using Nef polyhedra, and Fukuda [12] provided a polynomial algorithm for variable number of polytopes and dimensions; this method was implemented by Weibel [25]. The first among the exact implementations, which is known to achieve the best results, serves as our groundwork. Less effort was invested in efficient construction of Minkowski sums under rotation. A novel method has been recently proposed by Lien [20], this method is efficient, though it does not use exact geometric computation, and hence presumably cannot reveal all the combinatorially distinct sums.

Natural applications for our method are collision detection and answering proximity queries, while the polytopes can translate and rotate; see [22] for a survey on this field. Among the vast number of publications on this subject, we cite a small sample. The algorithm by Lin and Canny for collision detection and its optimized versions [9, 16, 21, 23] maintain the shortest distance between two polytopes and update it according to their movement. The algorithm by Gilbert, Johnson, and Keerthi [14] and its enhancement [5] obtain the shortest distance between the polytopes by using simplices from both polytopes. The algorithm by Kim and Rossignac approximate collisions under screw motions [4].

The rest of the paper is organized as follows. Section 2 reviews the spherical Gaussian map framework and the sqrt-extension exact number-type. Section 3 defines the criticality maps in one, two, and three dimensions. Section 4 describes two enhancements to the original algorithm for rotation about a general axis and for economical preprocessing of the criticality map. The outcome of various experiments is reported in Section 5. Directions for future work are suggested in Section 6.

2. PRELIMINARIES

2.1 The Spherical Gaussian Map

The spherical Gaussian-map of a polytope P in Euclidean three-dimensional space is the image of a set valued function from P to the unit sphere \mathbb{S}^2 , which maps each point p to a set of outward unit normals to support planes to P at p . Thus, the whole of a facet f of P is mapped to a single point on \mathbb{S}^2 — the outward unit normal to f . An edge of P is mapped to a geodesic segment on \mathbb{S}^2 . A vertex of P is mapped to a face in \mathbb{S}^2 . The above mapping is not injective, meaning that distinct polytopes are mapped to the same Gaussian map.

The map overlay of two subdivisions S_1 and S_2 , embedded on a surface Σ , is a subdivision S embedded on Σ , such that there is a face f in S iff there are faces f_1 and f_2 in S_1 and

S_2 , respectively, such that f is a maximal connected subset of $f_1 \cap f_2$. The overlay of the Gaussian maps of two polytopes identifies all pairs of features of both polytopes that have parallel supporting planes, as they occupy the same space on the unit sphere, thus, identifying all the pairwise features that contribute to the boundary of their Minkowski sum. Hence, Overlaying two Gaussian maps of two polytopes, respectively, results with the Gaussian map of the Minkowski sum of the two polytopes [11]

We need to obtain the Minkowski-sum polytope. Thus, we are interested in a map that preserves distinctness. To this end, we extend the notion of Gaussian maps. In particular, we extend the mapping of vertices. In the extended mapping each vertex v of P is mapped to a pair — a face in \mathbb{S}^2 and the vertex v itself. The extended mapping maps distinct polytopes to distinct extended Gaussian-maps. Gaussian maps refer to extended Gaussian-maps hereafter.

A Gaussian map can be represented as an arrangement of geodesic arcs embedded on a sphere. An extended Gaussian-map can be represented as an extended arrangement, where each face is extended with the coordinates of the corresponding polytope-vertex.

Our method computes the Minkowski sum in three-dimensional space using the (extended) Gaussian-map approach; see [11] for an overview about the subject. The overlay is computed by examining pairs of concurrent (either intersecting or incident) features one from each of the two Gaussian maps, according to the cases listed below. The exact method by which these pairs can be identified is discussed later. We briefly review the ten cases that may arise when the Gaussian maps of two polytopes P and Q are overlaid. We denote the Gaussian map of a polytope X as G_X . Let v_1 , e_1 , and f_1 denote a vertex, an edge, and a face of G_P and let v_2 , e_2 , and f_2 denote a vertex, an edge, and a face of G_Q , respectively. The ten cases follow.

1. A new vertex is induced by coinciding v_1 and v_2 .
2. A new vertex is induced by v_1 lying on e_2 .
3. A new vertex is induced by v_2 lying on e_1 .
4. A new vertex is induced by v_1 contained in f_2 .
5. A new vertex is induced by v_2 contained in f_1 .
6. A new vertex is induced by e_1 intersecting e_2 .
7. A new edge is induced by e_1 overlapping e_2 .
8. A new edge is induced by e_1 contained in f_2 .
9. A new edge is induced by e_2 contained in f_1 .
10. A new face is induced by f_1 overlapping f_2 .

Our implementation is based on CGAL [1]. In particular it uses the CGAL arrangement package [3, 27]. The package supports the construction, maintenance, and manipulation of arrangements of curves embedded on certain two-dimensional orientable parametric surfaces in three-dimensional space.¹ It also enables the extension of the arrangement cells (i.e., vertices, edges, and faces) with auxiliary data. The package supports, among the other, arrangements of geodesic arcs embedded on a sphere, and it contains a specific concretization for such arrangements, properly extended, that represents the Gaussian-map of a polytope [2]. As the package supports the overlay operation, computing the Gaussian-map of the Minkowski sum of two polytopes is immediately available.

2.2 Sphere Parameterization

¹Arrangement embedded on parametric surfaces are supported as of version 3.4, albeit only partially documented.

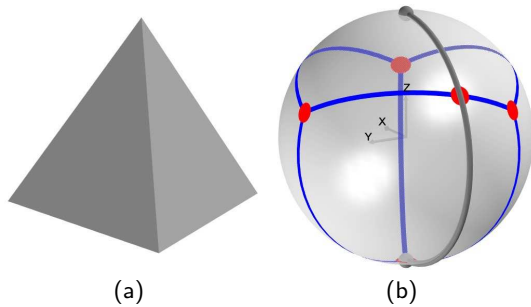


Figure 1: (a) A tetrahedron. (b) The arrangement that represents the Gaussian map of the tetrahedron. The contraction points are drawn as small balls and the identification curve is drawn as a thin tube.

A parametric surface S in three-dimensional space is given by a continuous function $\phi_S : \Phi \rightarrow \mathbb{R}^3$, where the domain $\Phi = U \times V$ is a rectangular two dimensional parameter space; $S = \phi_S(\Phi)$. U and V are open or closed intervals with endpoints in $\mathbb{R} \cup \{-\infty, +\infty\}$. We use the following parameterization of the sphere: $\Phi = [-\pi, \pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\phi_S(u, v) = (\cos u \cos v, \sin u \cos v, \sin v)$. This parameterization induces two contraction points $p_s = (0, 0, -1) = \phi_S(u, -\frac{\pi}{2})$ and $p_n = (0, 0, 1) = \phi_S(u, \frac{\pi}{2})$, referred to as the south and north poles, respectively, and an identification curve $\{\phi_S(\pi, v) \mid -\frac{\pi}{2} \leq v \leq \frac{\pi}{2}\}$, as $\phi_S(-\pi, v) = \phi_S(+\pi, v)$ for all v , which coincides with the opposite Prime (Greenwich) Meridian; see Figure 1b.

A curve γ is a continuous function $\gamma : I \rightarrow \Phi$, where I is a closed interval. A *u-monotone curve* is the image of a curve γ , such that if $t_1 < t_2$, then $u(\gamma(t_1)) < u(\gamma(t_2))$. A *vertical curve* is the image of a curve γ , such that $u(\gamma(t)) = c$ for all $t \in I$ and some $c \in U$. The three curves incident to the south pole in the Gaussian map depicted in Figure 1b are vertical. A weakly *u-monotone curve* is either vertical or *u-monotone*.

2.3 Arrangement Representation

We use a data structure that maintains arrangements embedded on a two-dimensional parametric surface in \mathbb{R}^3 to represent Gaussian maps of polytopes. This arrangement data-structure maintains the subdivision of the ambient space, a sphere in our case, into (i) cells of dimension 0 (*vertices*) embedded as points, (ii) cells of dimension 1 (*edges*) embedded as continuous weakly *u-monotone curves*, which are pairwise disjoint in their interiors, and (iii) cells of dimension 2 (*faces*). It provides the necessary capabilities for maintaining the embedded graph, while associating geometric data with the vertices, edges, and faces of the graph. The embedded graph is represented using a *doubly-connected edge list* (DCEL) [8, §2.2], which maintains the incidence relations on its cells [26] Each edge is represented by two halfedges with opposite orientations, and each halfedge is associated with the face to its left. Recall that a face is the mapping of a polytope vertex. Every face record of the DCEL is extended with the coordinates of the corresponding polytope vertex.

A vertex that coincides with one of the contraction points (the poles in our case), or lies on the identification curve (the opposite Prime Meridian in our case) is called a *boundary vertex*. The embedding point of a boundary vertex on

the sphere has more than one pre-image in the domain Φ , and thus must be handled in a special way. When a curve c is inserted into an arrangement, if c intersects the (closed) identification curve in its interior, c is split into two sub-curves at the intersection point, and the two subcurves are inserted into the arrangement. As a result, a new boundary vertex of degree two is created along with two new edges and vertices that correspond to the end points of c (assuming they did not exist in the arrangement before the insertion). Boundary vertices created as a result of a split of a curve as described above do not correspond to a face of a polytope. The arrangement depicted in Figure 1b contains four vertices of degree three each (one of them is located at the south pole), that correspond to the tetrahedron facets depicted in Figure 1a. The fifth vertex is a boundary vertex, which is a representation artifact.

2.4 The Sqrt Extension Number-Type

Square-root numbers have the form $a + b\sqrt{c}$, where a , b , and c are rational numbers. Each subset that has a particular c is closed under arithmetic operations and order relation. The rational number c is referred to as the extension. The sqrt-extension number-type of CGAL represents square-root numbers. It is equipped with the implementation of a set of arithmetic operations and order relations that exploit identical extensions of operands. It also provides the ability to compare two sqrt-extension numbers with different extensions efficiently.

The running times of the arithmetic operations and order relations provided by the sqrt extension when the identical-extension condition is met are comparable to the running times of the corresponding arithmetic operations of rational number-types. The running times of order relations when the condition is not met are only slightly larger. In practice, using number-types that represent (arbitrary) algebraic numbers increases the running times of the application significantly. Therefore, when all the non-rational values are square-root numbers representing them as sqrt-extension numbers is extremely advantageous.

3. CRITICALITY MAPS OF MINKOWSKI SUMS OF ROTATING POLYTOPES

3.1 Rotation About One Axis

Let P and Q be two polytopes with m and n vertices, respectively, and let M be their Minkowski sum, which is known to have at most $O(mn)$ vertices. Without loss of generality, we assume that P rotates counterclockwise about the z -axis, looking towards the origin from $z = \infty$, and Q is stationary. When rotating P and recomputing the Minkowski sum with Q , if the movement is sufficiently “small” (an exact definition for “small” is provided below), the combinatorial structure of the Minkowski sum does not change. Our work is based on this insight.

We discover the exact points where the combinatorial structure of the Minkowski sum of P and Q changes when P rotates. Those points, represented as pairs of sine and cosine of rotation angles, are stored in an efficient search-structure, which facilitates answering queries; see Section 3.1.5 for further details. We claim that between two critical points all the Minkowski sums of rotated P and stationary Q have the same combinatorial structure, while in two different sec-

tions bounded by critical points, the combinatorial structure is different.

The Minkowski sum of two polytopes is computed by converting their representation to Gaussian maps and overlaying their maps [11]. We denote a polytope P rotated by an angle θ as P^θ . Let M be the Minkowski sum of P and Q and let M_θ be the Minkowski sum of P^θ and stationary Q . As described above, there are ten structural cases, which are handled during the construction of the overlay. M and M_θ are combinatorially equivalent iff the sequence of handled cases is identical for the construction of G_M and G_{M_θ} , and combinatorially different otherwise. We calculate in advance the exact angles that cause the combinatorial structure of the Minkowski sum to change. These are the points where a new feature appears in or disappears from, the Minkowski sum structure.

3.1.1 The Criticality Map

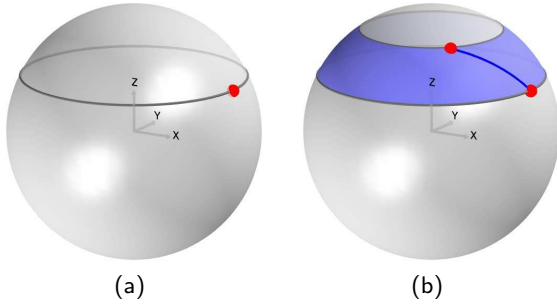


Figure 2: (a) A vertex rotation-circle. (b) An edge rotation-ring.

The input to our algorithm are polytopes P and Q represented as Gaussian maps. Given a vertex v on a Gaussian map, we define the *rotation plane* of v as the plane containing v and orthogonal to the rotation axis. We define the *rotation circle* of v as the intersection of the rotation plane of v and the sphere; see Figure 2a. Similarly, we define the *rotation volume* of an edge e as the volume bounded by the rotation planes of the endpoints of e . We define the *rotation ring* of e to be the intersection of the rotation volume of e and the sphere; see Figure 2b. We argue that all the critical points can be obtained by intersecting the rotation circles of the vertices of G_P with the edges of G_Q , and the rotation rings of G_P with the vertices of G_Q . In more details, for every vertex of G_P , we calculate its rotation circle and check whether it crosses an edge of G_Q . Then, for every edge in G_P , we calculate its rotation ring and check whether it contains a vertex of G_Q . According to the above claim the critical points are the intersections of the circles and rings of G_P with the curves and vertices of G_Q , respectively.

We are interested in the angles associated with the intersection points rather than the intersection points themselves. Observe that rotating an edge towards a vertex counter-clockwise results with the same angle as rotating the vertex towards the edge clockwise. Hence, due to implementation reasons, instead of calculating the intersections between the rotation rings of the edges of G_P and the vertices of G_Q , we calculate the intersections between the rotation circles of the vertices of G_Q and the edges of G_P .

It is not immediately clear why using the proposed algorithm and calculating the intersection points between vertices and edges ensures that all the critical points of the overlay are discovered. This is explained in the remainder

of this section by showing that our method predicts all the structural changes in the overlay map of G_P and G_Q , when P rotates and Q is stationary.

We go over the ten cases of the overlay as listed in Section 2.1 above. As for case 1, every vertex must be an endpoint of some edge, as there are no isolated vertices in the Gaussian map. Therefore, the intersections between all rotation circles of all vertices of G_P and all the edges of G_Q reveal all critical angles of this type. It is easy to see why changes in case 2 is discovered by intersecting the rotation circle of v_1 with e_2 . Case 3 is analogous. Case 4 is easily discovered, since changes in the topological structure of the overlay may happen only when v_1 rotates and intersects the bounding edges of the face. Case 5 is analogous.

Case 6 requires a little more attention. Intuitively, one can mistakenly think that the intersection point between the edges is a critical point, but since changing the intersection point itself, while the edges are still intersecting does not change the combinatorial structure. The only critical point occurs when one of the endpoints of the edges intersects the edge of the other. In case 7, it is clear that rotating an endpoint of e_1 and calculating where it intersects with e_2 results with the exact point of the structural change. In cases 8 and 9 when $e_i, i \in \{1, 2\}$ rotates and first touches the bounding edges of the face, the endpoints of the edge are the first to touch it. Hence, these cases are indirectly covered when handling cases 4 and 5, respectively. The last case 10 is also analogous to cases 4 and 5.

For every vertex v in G_P with Cartesian coordinates (x, y, z) we compute all the angles associated with the intersections between the rotation circle of v and the edges of G_Q . The angle computation is performed using simple trigonometric equations. Let e be an edge of G_Q . The embedding of e on the sphere is an arc of a great circle. It is contained in a plane p , which also contains the origin. Let (n_x, n_y, n_z) be the normal to p . Every great circular arc can be identified by the normal to the plane containing it, and its source and target vertices.

First, we calculate the intersection between the rotation circle of v and the plane p . The intersection point $v' = (x', y', z')$ must lie on the plane p ; thus, we get: $n_x \cdot x' + n_y \cdot y' + n_z \cdot z = 0$. Substituting the rotated coordinates of the intersection point $x' = x \cos \theta - y \sin \theta$, $y' = x \sin \theta + y \cos \theta$, and $z' = z$ yields the following system of two equations and two variables $\sin \theta$ and $\cos \theta$:

$$\cos^2 \theta + \sin^2 \theta = 1$$

$$n_x(x \cos \theta - y \sin \theta) + n_y(x \sin \theta + y \cos \theta) + n_z z = 0.$$

Solving the system results with the following closed forms:

$$\sin \theta = -\frac{(a \cdot c)}{a^2 + b^2} \pm \frac{b \cdot \sqrt{a^2 + b^2 - c^2}}{a^2 + b^2} \quad (1)$$

$$\cos \theta = -\frac{(b \cdot c)}{a^2 + b^2} \pm \frac{a \cdot \sqrt{a^2 + b^2 - c^2}}{a^2 + b^2}, \quad (2)$$

where $a = (x \cdot n_y - n_x \cdot y)$, $b = (x \cdot n_x - n_y \cdot y)$, and $c = z \cdot n_z$.

The same procedure is applied to the vertices of G_Q with the edges of G_P using clockwise rotation, and the critical points from the two steps are merged into one list. The calculations for the clockwise rotations are similar.

The quadratic equations may have zero, one, or two solutions. Degenerate cases may occur. When the rotation circle of v does not intersect p , there is nothing to do. A

single intersection point is handled properly. A rotation circle that lies on p constitutes another degenerate case. The critical points in this case are the endpoints of e . Since there are no isolated vertices, these points are also the endpoints of other edges. The corresponding critical points are discovered when those other edges are processed (together with v). Thus, this case can be ignored.

The process is applied to all the vertices of G_P and G_Q , except for vertices located at the poles, as these vertices do not change their position while rotating the polytope about the z axis. In addition, recall that when inserting an edge which intersects the identification curve, the edge is split into two edges (see Figure 1b), and an additional vertex is added at the intersection point. Since this vertex is created for technical reasons and is not associated with a real polytope facet, the process is not applied to it.

Solving the equation system described above yields the exact angles associated with intersection points of rotation circles of vertices of one Gaussian map and the planes that contain the edges of the other Gaussian map. We need to exclude angles associated with intersections that do not lie on the corresponding edges. In order to decide whether the intersection point lies on the edge, we project the vertex point and the edge curve onto the xy -plane and check whether the projected point lies in the projected curve. Degenerate cases, such as the case where the edge has a vertical curve (see section 2.2) are correctly handled. In such cases we project the vertex point and the edge curve onto the zx -plane instead and similarly check whether the projected point lies in the projected curve.

Table 1: Complexities of polytopes and Gaussian-map representations. Icos. — Icosahedron, DP — Dioctagonal Pyramid, PH — Pentagonal Hexecontahedron, TI — Truncated Icosidodecahedron, GS4 — Geodesic sphere level 4, HE — Number of halfedges.

Object Type	Polytope			Gaussian Map		
	V	E	F	V	HE	F
Icos.	12	30	20	72	192	36
DP	17	32	17	105	304	59
PH	92	150	60	196	684	158
TI	120	180	62	230	840	202
GS4	252	750	500	708	2124	366

Table 2: Complexities of the primal representations of Minkowski sums and numbers of *critical changes* and critical points. ODP — Orthogonal Diocagonal Pyramid, RGS4 — Rotated Geodesic sphere level 4; see Table 1 for other abbreviations.

Smd 1	Smd 2	Primal			critical changes	critical points
		V	E	F		
Icos.	Icos.	65	110	47	241	221
DP	ODP	132	261	132	451	286
PH	TI	285	483	200	2262	1761
GS4	RGS4	1048	2582	1536	27041	12511

The criticality map, which is the result of the above process, is a one-dimensional arrangement of angles. For two arrangements of m and n vertices, (and $O(m)$, $O(n)$ edges respectively), the upper bound for the complexity of the resulting criticality map is obviously $O(mn)$. Table 2 lists the actual number of critical points for selected polytopes (the examples are taken from [11]). We present both the number of *critical changes*, which is the number of pairs of intersect-

ing rotation circles and edges, and the number of *critical points*, which is the number of different critical angles.

3.1.2 Number Type

We exploit the fact that the equations above are quadratic, and use the sqrt-extension number-type of CGAL; see Section 2.4. Thus, combining exact results with efficient running times for the algorithm.

Each critical point is represented as a pair of sine and cosine, which are represented in turn as sqrt-extension numbers. Note that for a fixed critical point the sine and cosine have the same extension. We use the efficient arithmetic operations provided by the sqrt-extension number-type to check whether the intersection of the rotation circle with the plane that contain an edge lies on the edge; see Section 3.1.1.

The critical points are stored in an efficient search-structure, and therefore must be compared. The sine and cosine of different critical points are likely to have different extensions. We exploit the comparison method provided by the sqrt-extension number type, which is much more efficient than the comparison method provided by arbitrary algebraic number-types.

3.1.3 Efficient Data-Structure

Recall that every section between two critical points is considered a cell, where all the Minkowski sums corresponding to rotation angles (of P) in that cell share the same combinatorial structure. We produce, for every cell a representative combinatorial structure.

The rational-rotation method [6], implemented in CGAL, accepts an angle α and an approximation bound δ and returns a rational sine and cosine of an angle, such that its distance from α is less than δ . For every cell we compute the middle angle and use the rational-rotation method to find β , an adjacent angle with rational sine and cosine. Then, we confirm that β is located inside the cell by comparing it to the critical points bounding the cell. If β is located outside the cell, we recalculate it by using the rational-rotation method with a tighter approximation-bound.

A key of the search structure is the critical-point angle in sqrt-extension exact number-type. Each entry in the search structure represents a critical point and contains a list of pairs of vertex and edge or edge and vertex that generated this point. In addition, for every cell we calculate the Minkowski sum of P^β using the *Arrangement-on-a-Sphere rotation algorithm* (see Section 3.1.4) and Q and maintain it along with the angle β , as part of the entry of its larger-angle critical point.

3.1.4 Arrangement-on-a-Sphere Rotation Algorithm

When discussing rotation of polytopes represented as Gaussian maps, rotating the arrangement representation instead of rotating the polytope and recomputing its Gaussian map is desired. When a polytope rotates its coordinates change, but its combinatorial structure persists. As the Gaussian map of a polytope reflects its structure, the incidence relations between the vertices, edges, and faces of the arrangement representing the Gaussian map do not change either. When rotating the Gaussian map the geometric data of the arrangement cells and the geometric data attached to the faces of the arrangement, namely the polytope vertex coordinates (see Section 2.1) must be updated. In addition, the rotation of the identification curve (as part of the rotation

of the entire embedding sphere) must be handled properly, as its location is immutable on the sphere. While this is a technical problem related to the specific implementation that we use, we believe similar problems would arise in other implementations as well. Curves that were split by the identification curve before the rotation may be split by the identification curve at a different intersection point or not intersect it at all after the rotation. Curves that did not intersect the identification curve before the rotation may intersect it after the rotation. Handling these structural changes correctly produces a valid rotated arrangement; see [2] for a detailed description of the arrangement structure.

The input to the rotation algorithm is an arrangement A of geodesic arcs embedded on a sphere and a rotation matrix R . This algorithm consists of several steps. First, we merge every pair of subcurves that are the result of splitting a curve at the identification curve when the curve was inserted into A ; see Section 2.3. When merging subcurves, we delete the corresponding intersection vertices from the boundary-vertices list and update the DCEL accordingly. Then, we rotate A by applying the rotation matrix R to the geometric data of the arrangement cells and their extended data. After moving the vertices, edges, and faces to their rotated location, we calculate whether the rotated curves intersect the identification curve. If an intersection is discovered, the edge is split into two edges. The newly created intersection vertex is added to the boundary-vertex list.

3.1.5 Answering a Query

Given a rotation angle θ , we wish to rapidly report the Minkowski sum of P^θ and Q . The method operates as follows. The user enters two polytopes one rotatable and one stationary. In the preprocessing phase the polytope criticality map is constructed. The query input is a series of pairs of rational sine and cosine values representing an exact rotation angle for the rotatable polytope. The output is the Minkowski sums of the rotatable polytope rotated by the input angles with the stationary polytope.

First, the cell containing the query angle is searched for in the data structure (in logarithmic time), and its entry is obtained. Each cell contains the exact rational sine and cosine of a representative angle θ' in that cell and the Gaussian map $G_{M_{\theta'}}$ of the Minkowski sum $M_{\theta'} = P^{\theta'} \oplus Q$. According to our claim, $M_{\theta'}$ and the sought polytope M_θ share the same combinatorial structure. We need to adjust the vertices of $M_{\theta'}$ attached to the faces of $G_{M_{\theta'}}$ to obtain the vertices of M_θ to compensate for the difference $\theta - \theta'$. Recall, that every vertex of M_θ is a sum of a vertex of P^θ and a vertex of stationary Q . We further extend the face of the DCEL (of Gaussian maps of Minkowski sums) to contain the original-polytope vertex of the rotated summand (P in our case) in addition to the vertex of the polytope; see Section 2.1 (the vertex of Q can be calculated directly). For every face of $G_{M_{\theta'}}$, we obtain the original vertex of P and the vertex of $M_{\theta'}$, extract the vertex of Q , and calculate the final vertex of M_θ . We can also update $G_{M_{\theta'}}$ with the vertices of M_θ ; however, if we do, the resulting arrangement-on-a-sphere becomes an invalid Gaussian map, unless we also update the geometric data of the arrangement points and curves during the final adjustment step. In summary, we can easily reconstruct the sought polytope M_θ using the polytope vertices stored in the arrangement face-records and the combinatorial structure encoded by the arrangement DCEL.

The Gaussian map of a polytope is a convex subdivision. Computing the overlay of such special subdivisions optimally can be done in linear time in the output size [10]. Thus, the Minkowski sum of two convex polytopes with m and n features, respectively, can be computed in $O(mn)$ time in the worst case. These bounds also apply to an arbitrary rotation axis as discussed in section 4.1 and are summarized in the following theorem:

THEOREM 1. *Given two polytopes P and Q with m and n vertices, respectively, P is rotating about an arbitrary axis, and Q is stationary, the criticality map can be computed in $O(mn \log(mn))$ time. Computing all the combinatorially different Minkowski sums can be done in $O(mn(mn + \Psi))$ time and $O(m^2n^2)$ space, where Ψ is the maximum time required to find an angle with rational sine and cosine inside a single cell of the criticality map. Retrieving the combinatorial structure for a certain rotation angle can be done in $O(\log(mn))$ time.*

In our implementation we do not exploit the optimal computation of the overlay of Gaussian maps. Instead, we resort to a ready made sweep-line based algorithm that exhibits good practical performance, and incurs a mere logarithmic factor over the optimal asymptotic computing-time.

3.2 Rotation About Two Axes

Let P and Q be two polytopes. Again we assume that Q is stationary and P can rotate, this time with *two* degrees of freedom. First, it rotates about the z -axis by θ , $-\pi \leq \theta \leq \pi$, and then about the new x -axis by ϕ , $-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$. Similar to the case of rotation about one axis, we denote such a rotated polytope by $P^{(\theta, \phi)}$. There are various ways to define two degrees of freedom of rotation. We chose this one since (i) the rotation space (θ, ϕ) is naturally embedded on a sphere, and (ii) we have come across mechanisms that have exactly these two degrees of freedom.

3.2.1 The Criticality Map

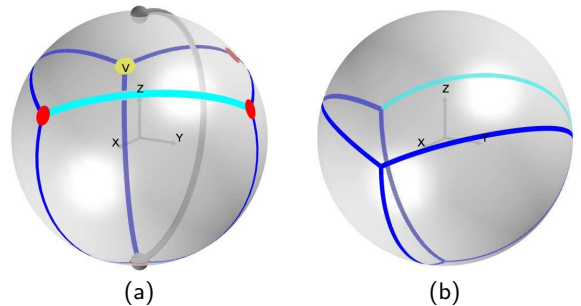


Figure 3: A subset of the criticality map of two identical tetrahedra for two axes rotation. (a) The Gaussian map of the tetrahedron depicted in Figure 1 viewed from a different point of view. (b) The critical curves generated by v and all the edges in the Gaussian map.

While the criticality map of rotation about one axis can be represented as a one-dimensional arrangement of critical points, the criticality map of rotation about two axes can be represented as an arrangement of critical curves on a sphere. A point (θ, ϕ) in polar coordinates on the criticality map represents $G_{P^{(\theta, \phi)}}$.

Every pair of vertex v in G_P and edge e in G_Q may generate a curve $\gamma : I \rightarrow \Phi$ (see Section 2.2) inducing the arrangement of criticality map for rotation about two axes. A

point on the curve $\gamma(t) = (\theta(t'), \phi(t'))$ for some $t' \in [0, 1]$ represents a rotation as described above, such that the vertex v of the rotated G_P lies on the edge e of Q ; see Figure 3. In a degenerate case a pair of vertex v in G_P and edge e in G_Q may generate an isolated critical point.

The curves of the criticality map represent the critical changes in the combinatorial structure of the Minkowski sum. All possible angle pairs represented as points contained in a face of the criticality map are associated with Minkowski sums that have the same combinatorial structure. Thus, for each face it is sufficient to store the Minkowski sum associated with a representative pair of angles represented by a point inside the face. For every face we find a point (θ, ϕ) located at the interior of the face, such that the sine and cosine of θ and ϕ are rational numbers. The Minkowski sum $P^{(\theta, \phi)} \oplus Q$ is computed and stored in the extended face.

We argue that the upper bound on the combinatorial complexity of the criticality map for rotation about two axes is $O(m^2n^2)$, when P and Q have m and n vertices, respectively. Notice that here we have a two-dimensional arrangement of $O(mn)$ well-behaved curves. This bound is tight; namely, there exist two polytopes, such that when one has two degrees of freedom of rotation as above, the polytopes induce $\Omega(m^2n^2)$ combinatorially distinct Minkowski sums. We skip the details here and refer the reader to the analogous, and more detailed, discussion of the case of rotation about three axes in Section 3.3.

The criticality map for rotation about two axes was not implemented in software though the technical details are defined as described above. The critical curves can be represented as well behaved algebraic curves. The current software available in CGAL arrangements handles only curves that are part of great circles over the sphere.

3.2.2 Answering a Query

Given a pair of angles (θ, ϕ) we wish to report the Minkowski sum $P^{(\theta, \phi)} \oplus Q$. First, using some point location strategy we find the face containing (θ, ϕ) in the criticality map. We use the Minkowski sum M' stored in the record of this face as the basis for the sought Minkowski sum. Let θ' and ϕ' be the rotation angles that correspond to M' and are also stored along M' . We adjust the final result by applying a rotation by the angle differences $\theta - \theta'$ and $\phi - \phi'$ to obtain the desired Minkowski sum.

3.3 Rotation About Three Axes

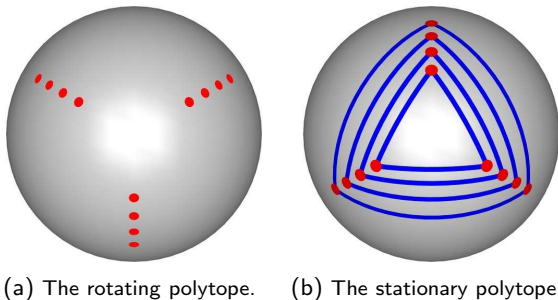


Figure 4: Lower bound $\Omega(m^3n^3)$ on the number of distinct Minkowski sums, when one polytope has three dofs of rotations.

The ultimate goal would be to compute the Minkowski-sum criticality-map in the most general case, that is, when one polytope has three degrees of freedom of rotation. There

are different ways to describe the full freedom of rotation of a rigid body in space; see, e.g., [7, Appendix E]. Typically, this involves three parameters, which we denote by α, β , and γ , without committing to a specific representation.

The criticality map in this case is three dimensional in the (α, β, γ) -space. A critical two-dimensional surface is determined by the set of (α, β, γ) -rotations for which a specific vertex of G_P intersects a specific edge of G_Q or vice versa. The total number of such surfaces is obviously $O(mn)$, each of which can be represented as an algebraic surface patch of constant maximum algebraic degree, and bounded by a constant number of curves, each of constant maximum algebraic degree as well, by using standard transformations. Thus, the overall number of cells in the underlying arrangement is at most $O(m^3n^3)$.

Interestingly, this bound can be attained. Namely, there are two polytopes with m and n vertices, respectively, such that they induce $\Omega(m^3n^3)$ combinatorially distinct Minkowski-sums when P rotates with three degrees of freedom.² The polytope Q is constructed as follows. We take a tetrahedron, align it, such that one of its facets, referred to as the base facet, becomes coplanar with the base of a hemisphere, and clip it several times so that the resulting object is contained in the hemisphere, and the Gaussian map of the resulting object includes a sequence of shrinking spherical triangles, as depicted in Figure 4b. We denote the three sets of edges on the three different sides of all triangles, respectively, by Γ_1, Γ_2 , and Γ_3 . The number of edges in each set is a sixth of the number of edges in G_Q , that is, $O(n)$, where n is the total number of cells in G_Q . The polytope Q is similarly constructed, but differently aligned. Figure 4a depicts the vertices of G_P relevant to the construction of the distinct Minkowski sums, where P is rotated 180° about the axis orthogonal to the base facet. We denote the three sets of vertices on the three different corners of the triangles, respectively, by Δ_1, Δ_2 , and Δ_3 . The number of vertices in each set is roughly a third of the number of vertices in G_P (There is one vertex, which is the mapping of the base facet, is not depicted.), that is, $O(m)$, where m is the total number of cells in G_P . It is possible to rotate P to yield $P^{(\alpha, \beta, \gamma)}$, such that for every choice of one vertex of each Δ_i for $i = 1, 2, 3$, and one pair of consecutive edges of Γ_i for $i = 1, 2, 3$, we can place the vertex selected from Δ_i in between the edges selected from Γ_i , for $i = 1, 2, 3$ in the overlay of $G_{P^{(\alpha, \beta, \gamma)}}$ and G_Q . We get a Minkowski sum, which is distinct from the sum we would get for any other selection. The total number of distinct sums is therefore $\Omega(m^3n^3)$.

Notice that in the case of three degrees of freedom of rotation, both the algebra, as well as the data-structure support for constructing such arrangements are far from trivial. We do not, at the moment, have sufficient infrastructure to support such constructions.

THEOREM 2. *Given two polytopes P and Q with m and n vertices, respectively, P is rotating about three axes, and Q is stationary, the number of distinct Minkowski sums of P and Q is at most $\Theta(m^3n^3)$. This bound is tight.*

4. ENHANCEMENTS

²The construction is an adaptation of a lower bound construction on the complexity of the configuration space of a polygon translating and rotating among point obstacles in the plane [18, Figure 1].

The algorithm described in Section 3 performs well on examples with small numbers of criticalities. When the number of criticalities increases the query time remains almost unchanged, but the preprocessing time and space dramatically increases; see Table 3. In addition, (general) rotation about three axes is computationally limited; see Section 3.3. The following sections describe two enhancements applied to the original method designed to handle these problems.

4.1 Rotation About an Arbitrary Axis

Real world applications are usually concerned with general rotation queries. While it is possible to answer general rotation queries using a rotation representation of three angles (about three axes, respectively), our results show that even for polytopes with a small number of vertices it is impractical. In order to answer a general rotation query it is better to use an axis-angle representation instead of using the three-angle representation. The axis-angle representation consists of a directed axis and an angle of rotation. Using this representation, producing all the Minkowski sums about a general rotation axis becomes feasible.

Given two polytopes P and Q and a rotation axis (u, v, w) (rational and normalized) we obtain all the critical points where the combinatorial structure of the Minkowski sum of P and Q changes while P rotates about the given axis. We compute the critical points using the same approach used in the single fixed-axis case; see Section 3. The complexity of the criticality map remains $O(mn)$ and the time to answer each query is the same as in the one fixed-axis case. However, the algebra in this case is a bit more complicated.

Let $v = (x, y, z)$ be a vertex of G_P and e an edge of G_Q , and let $N_p = (n_x, n_y, n_z)$ be the normal to the defining plane p of e . First, we need to calculate the intersection $v' = (x', y', z')$ between the rotation circle of v and p . Let R be the matrix for rotation by an angle of θ about (u, v, w) .³ For convenience, we define $f_\theta = (1 - \cos \theta)$ and get:

$$R = \begin{pmatrix} u^2 f_\theta + \cos \theta & uv f_\theta - w \sin \theta & uw f_\theta + v \sin \theta \\ uv f_\theta + w \sin \theta & v^2 f_\theta + \cos \theta & vw f_\theta - u \sin \theta \\ uw f_\theta - v \sin \theta & vw f_\theta + u \sin \theta & w^2 f_\theta + \cos \theta \end{pmatrix}$$

v' lies on the plane p ; therefore, $\langle N_p, v' \rangle = 0$. Substituting v' with Rv yields Equations 1 and 2, where

$$\begin{aligned} a &= x(n_y w - n_z v) + y(n_z u - n_x w) + z(n_x v - n_y u), \\ b &= xn_x(1 - u^2) - xu(n_y v + n_z w) - yv(n_x u - n_z w) \\ &\quad + yn_y(1 - v^2) - zw(n_y v - n_x u) + zn_z(1 - w^2), \\ c &= xu(n_x u + n_y v + n_z w) + yv(n_x u + n_y v + n_z w) \\ &\quad + zw(n_x u + n_y v + n_z w). \end{aligned}$$

The remaining steps are identical to the corresponding steps in the single fixed-axis case; see Section 3.

4.2 A Dynamic Approach

Since already for polytopes with a moderate number of vertices, retaining all Minkowski sums is infeasible, we now describe a method that balances between the query and preprocessing time.

Recall, that all critical points are determined by the intersection between the rotation circles of vertices of one Gaus-

³See, for example, http://en.wikipedia.org/wiki/Rotation_matrix.

sian map and the edges of the other, and vice versa. The combinatorial structures of the Minkowski sum of two adjacent cells are similar except for a typically small number of local changes. It is more space and time efficient to encode the local changes and store them at the critical points, instead of constructing and storing the entire combinatorial structure of the sum in every cell. It is still possible to construct the correct structure of the sum in some cell by applying the local changes stored in an incident critical point, on the structure of its appropriate adjacent cell.

Calculating only a small portion of the sums reduces the preprocessing time and space dramatically. On the other hand, applying structural changes increases the query time. The user can balance between the two providing a real parameter $\rho, 0 < \rho < 1$, which determines the ratio between the number of cells that are preprocessed and the total number of cells. The preprocessed cells are selected uniformly. They are referred to as the *construction cells*. After the preprocessing step every cell contains the $\sin \theta$ and $\cos \theta$ pair of an angle θ contained in the cell, such that the pair of values are rational. Only construction cells contain in addition the Minkowski sum between Q and P^θ .

Assume that a user issues a query with the angle θ seeking M_θ , the Minkowski sum of P^θ and Q . Let C denote the cell containing θ . We easily find C in the criticality map. If C is a construction cell, we continue according to the original algorithm. Otherwise, we traverse the criticality map searching for the nearest construction cell, as we would like to perform as little as possible number of structural updates. We traverse the discovered cells in reverse order applying the changes encoded in the critical points incident to the cells in the order of traversal.

Let C_1 and C_2 be two adjacent cells traversed during the update process storing the angles θ_1 and θ_2 , respectively, and let i be the critical point incident to both C_1 and C_2 . Assume that C_1 is either a construction cell storing the Minkowski sum $M_{\theta_1} = P^{\theta_1} \oplus Q$, or M_{θ_1} has already been reconstructed as part of the update process. The combinatorial structure of M_{θ_1} and $M_{\theta_2} = P^{\theta_2} \oplus Q$, differ only by the changes induced by the vertex (or vertices) and the edge (or edges) that generated the critical point i .

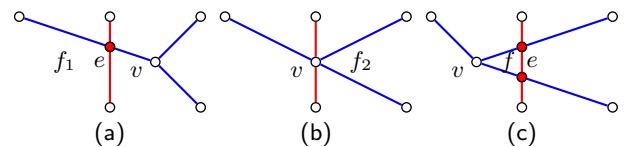


Figure 5: (a) Before the critical point — v is located on the right-hand side of e . (b) At the critical point — v lies on e . (c) After the critical point — v is located on the left-hand side of e .

When a critical point is generated by a vertex of G_P and an edge of G_Q the vertex v of G_P is located on the right-hand side of the edge e of G_Q before the change (see Figure 5a), and on the left-hand side of e after the change (see Figure 5c). The Minkowski sums M_{θ_1} and M_{θ_2} reflect the relative placements of v and e . The update is performed in two steps: First, all the incident edges to v that are split by e are merged. Next, all the incident edges of v that were not split by e are split. The opposite case is analogous.

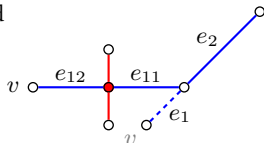
In the original algorithm we produce the correct Gaussian maps that represent the rotated polytopes, and then compute the corresponding Minkowski sums. However, when

Table 3: Time consumption (in seconds) of Minkowski sum computation. Icos. – Icosahedron, DP – Diocagonal Pyramid, ODP – Orthogonal Diocagonal Pyramid, PH – Pentagonal Hexecontahedron, TI – Truncated Icosidodecahedron. Sgm – Spherical Gaussian map method [11], Criticality Map 1 – Full criticality map, Criticality Map 2 – Dynamic criticality map preprocessing half of the critical points, Criticality Map 3 – Dynamic criticality map preprocessing one fifth of the critical points, Preprocessing – Preprocessing time, Query – Query time

Smd 1	Smd 2	Sgm	Crit. Map 1 ($\rho = 1$)		Crit. Map 2 ($\rho = 0.5$)		Crit. Map 3 ($\rho = 0.2$)	
			Preprocessing	Query	Preprocessing	Query	Preprocessing	Query
Icos.	Icos.	0.02	4.09	<0.01	3.47	0.01	2.06	0.01
DP	ODP	0.03	8.97	<0.01	5.52	0.01	3.41	0.02
PH	TI	0.03	127.07	0.02	66.08	0.02	29.15	0.04

we update the Minkowski sum while applying a change, encoded in a critical point, to the arrangement representing the Gaussian map of the Minkowski sum, we only maintain the correct combinatorial structure of the Minkowski-sum. This produces a temporary valid arrangement that does not however represent a valid Gaussian map any longer. Consider the change caused by the vertex-edge case depicted in Figure 5. We relocate the point associated with the vertex v and update the coefficients of the curves associated with edges incident to v , but we do not update other vertices and edges that do not participate in the local change saving significant processing-resources. To this end, every vertex maintains its original point (the point of the Minkowski sum from the construction cell), every edge maintains its original curve that induces it, and every face maintains handles to the two faces, one of G_P and of G_Q , that induce it. Transforming the original points and curves according to the current rotation angle results with the valid structure for that rotation. The geometric data of the polytope vertices of the Minkowski-sum is calculated using the handles maintained at the faces.

Consider again the change caused by the update step depicted in the figure to the right. The underlying great circle of the edge e_1 changes and it is split into e_{11} and e_{12} . Suppose that e_1 and e_2 are the result of a split performed as part of an update previously applied, and that at some point e_{11} and e_2 become the operands of a merge. Two geodesic arcs can be merged only if they share an endpoint and they lie on the same great circle. The merge operation is performed only if this precondition is met. At this point, the two subcurves are topologically mergeable, but geometrically they violate the geometric precondition, presented above, required for them to be merged. Hence, we had to suppress the preconditions by re-implementing the merge operation.



As mentioned above, every face is extended with the coordinates of the corresponding polytope-vertex. Every vertex in the Minkowski sum is the sum of two vertices — one from P and another from Q . During the structural change the Minkowski-sum vertices are copied from the faces of $G_{M_{\theta_1}}$ to the faces of $G_{M_{\theta_2}}$. When new faces are created their handles to the faces of the summands are updated. The Minkowski-sum vertices are calculated from the vertices of the summands stored in the records of the corresponding faces of the Gaussian maps.

New faces are created iff the number of halfedges incident to v intersecting e in $G_{M_{\theta_1}}$ is smaller than the number of halfedges incident to v intersecting e in $G_{M_{\theta_2}}$; see Figure 5 for an illustration of the case where a new face f is created. When a new face is created we obtain the handles to the

two faces, say f_P and f_Q , of the Gaussian maps of the two summands that induce the new face and store them in the record of the newly created face. This is easily done using the incidence relations maintained by the DCEL and the extra face handles stored in each face record. In the figure the handle to f_P is stored in the record of f_1 and the handle to f_Q is stored in the record of f_2 .

THEOREM 3. *Given a real number ρ , $0 < \rho \leq 1$, and two polytopes P and Q with m and n vertices, respectively, P is rotating about an arbitrary axis, and Q is stationary, it is possible to compute a partial criticality map that contains only $O(\lceil \rho mn \rceil mn)$ distinct Minkowski sums in $O(\lceil \rho mn \rceil (mn + \Psi))$ time and $O(\lceil \rho mn \rceil mn)$ space, where Ψ is the maximum time required to find an angle with rational sine and cosine inside a single cell of the criticality map. Then, obtaining the Minkowski sum for a certain rotation angle can be done in $O(\log(mn) + \chi/\rho)$ time, where χ is the maximum time complexity of the handling of a single critical point in general position.*

5. EXPERIMENTS

We are not aware of any previous work that updates the exact Minkowski sum of rotating polytopes. Recently, Lien [20] suggested an algorithm to handle Minkowski sums of rotating polytopes, calculating their Minkowski sum by updating the orientation of each feature. As Lien’s method does not use exact geometric computation, it presumably cannot identify all possible distinct structures that arise during the rotation; therefore, we believe comparing our method with Lien’s is unavailing. For the exactness of our algorithm we pay with longer running times. At the same time, when allowing for degeneracies and hence dealing with delicate situations, exactness is mandatory.

The results reported in Table 3, address the following question: Given two polytopes P and Q , a rotation axis r (rational and normalized) and an angle α , compute the Minkowski sum of P rotated about r by α with Q . We compare the results of four different solutions. The first was obtained by the static exact algorithm by Fogel and Halperin [11]; it is denoted **Sgm** in the table: To answer this query using **Sgm**, we rotate P about r using the arrangement-on-sphere rotation algorithm (see Section 3.1.4) and reconstruct the Minkowski sum of the P^α and Q . **Critically Map 1, 2, and 3** refer to executions of our dynamic algorithm with different parameters. The two sub-columns list the time consumed by the preprocessing and query stages, respectively, for the corresponding execution. Critically Map 1 refers to the execution that retains all the Minkowski sums about r in the preprocessing stage. Critically Map 2 and 3 refer to the executions that retain only 50% and 20% of the Minkowski sums, respectively.

Table 4: Time consumption (in seconds) of computing all the possible different structures of Minkowski sum about one fixed axis. GS4 — Geodesic Sphere Level 4, RGS4 — Rotated Geodesic Sphere Level 4, Sgm — Rotating Smd 1 and computing the Minkowski sum with Smd 2 using Sgm. See Table 3 for other abbreviations.

Smd 1	Smd 2	Critical Points	Critically Map	Sgm
Icos.	Icos.	221	0.69	2.47
DP	ODP	286	1.93	5.02
PH	TI	1761	21.32	84.91
GS4	Icos.	1620	50.27	243.65
Icos.	RGS4	2341	63.34	204.11

Table 4 deals with the series of all the possible combinatorially distinct Minkowski sum structures while rotating one polytope about a fixed axis. For both compared methods, we first calculate the criticality map and build a list of queries where every query is concerned with a combinatorially distinct Minkowski sum. Using the criticality map method, we query the search structure for the requested Minkowski sum. Using the Sgm method, we rotate the rotatable polytope for every query and calculate from scratch its Minkowski sum with the stationary polytope.

6. FUTURE WORK

This work was implemented as part of a complete integrated framework for proximity queries using the spherical Gaussian map. We plan to design another approach for solving rotation problem in collision detection, by implementing an algorithm in the spirit of Lin and Canny [21], in order to complete this framework. For every pair of polytopes we can find the closest features among them, and calculate only a small portion of the Minkowski sum including those features. This Minkowski sum is updated according to the movement of the polytopes.

Another direction for future research is creating an implementation of the two-axis criticality map. The theoretical base for the solution is described in Section 3.2.1. Once we support more general algebraic curves on a sphere, implementing the two-axis rotation algorithm will be feasible.

7. REFERENCES

- [1] CGAL *User and Reference Manual*, 3.6 edition, 2010. http://www.cgal.org/Manual/last/doc_html/cgal_manual/packages.html.
- [2] E. Berberich, E. Fogel, D. Halperin, M. Kerber, and O. Setter. Arrangements on parametric surfaces II: Concretizations and applications. *To appear in Math. in Comput. Sci.*, 2010.
- [3] E. Berberich, E. Fogel, D. Halperin, K. Melhorn, , and R. Wein. Arrangements on parametric surfaces I: General framework and infrastructure. *To appear in Math. in Comput. Sci.*, 2010.
- [4] K. Byungmoon and R. Jarek. Collision prediction for polyhedra under screw motions. In *Proc. 18th ACM Symp. Solid Phys. Model.*, pages 4–10. ACM Press, 2003.
- [5] S. A. Cameron. Enhancing GJK: computing minimum and penetration distances between convex polyhedra. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 3112–3117, Apr 1997.
- [6] J. Canny, B. Donald, and E. K. Ressler. A rational rotation method for robust geometric algorithms. In *Proc. 8th Annu. ACM Symp. Comput. Geom.*, pages 251–260. ACM Press, 1992.
- [7] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion*. MIT Press, 2005.
- [8] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, Germany, 3rd edition, 2008.
- [9] S. A. Ehmman and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. 20(3):500–510, 2002. Proc. of Eurographics’2001.
- [10] U. Finke and K. H. Hinrichs. Overlaying simply connected planar subdivisions in linear time. In *Proc. 11th Annu. ACM Symp. Comput. Geom.*, pages 119–126. ACM Press, 1995.
- [11] E. Fogel and D. Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. *Comput. Aided Design*, 39(11):929–940, 2007.
- [12] K. Fukuda. From the zonotope construction to the Minkowski addition of convex polytopes. *J. of Symbolic Computation*, 38(4):1261–1272, 2004.
- [13] P. K. Ghosh. A unified computational framework for Minkowski operations. *Comput. & Graphics*, 17(4):357–378, 1993.
- [14] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects. *IEEE Trans. on Robotics and Automation*, 4(2):193–203, 1988.
- [15] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: Computational complexity and applications to Gröbner bases. *SIAM J. on Disc. Math*, 6(2):246–269, 1993.
- [16] L. J. Guibas, D. Hsu, and L. Zhang. H-walk: Hierarchical distance computation for moving convex bodies. In *Proc. 15th Annu. ACM Symp. Comput. Geom.*, pages 265–273, 1999.
- [17] P. Hachenberger. Exact Minkowski sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces. In *Proc. 15th Annu. Eur. Symp. Alg.*, volume 4698 of *LNCS*, pages 669–680. Springer, 2007.
- [18] D. Halperin and M. Sharir. A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. *Disc. Comput. Geom.*, 16:121–134, 1996.
- [19] A. Kaul and J. Rossignac. Solid-interpolating deformations: Construction and animation of pips. *Comput. & Graphics*, 16(1):107 – 115, 1992.
- [20] J.-M. Lien. Movie: Minkowski sums of rotating convex polyhedra. In *Proc. 24th Annu. ACM Symp. Comput. Geom.*, pages 228–229. ACM Press, 2008.
- [21] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1008–1014, 1991.
- [22] M. C. Lin and D. Manocha. Collision and proximity queries. In J. E. Goodman and J. O’Rourke, editors, *Handb. Disc. Comput. Geom.*, chapter 35, pages 787–807. Chapman & Hall/CRC, Boca Raton, FL, 2nd edition, 2004.
- [23] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Trans. Graph.*, 17(3):177–208, 1998.
- [24] G. Varadhan and D. Manocha. Accurate Minkowski sum approximation of polyhedral models. In *Proc. Pacific Conf. on Comput. Graphics and Appl.*, pages 392–401. IEEE Comput. Society Press, 2004.
- [25] C. Weibel. Minkowski sums. <http://roso.epfl.ch/cw/poly/public.php>.
- [26] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. Advanced programming techniques applied to CGAL’s arrangement package. *Comput. Geom. Theory Appl.*, 38(1–2):37–63, 2007. Special issue on CGAL.
- [27] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. 2D arrangements. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.6 edition, 2010.