Computer-Aided Design 43 (2011) 1258–1269

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Fast and robust retrieval of Minkowski sums of rotating convex polyhedra in 3-space[☆]

Naama Mayer, Efi Fogel, Dan Halperin[*]

The Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

ARTICLE INFO

Keywords:
Minkowski sums
Convex polytopes
Collision detection
Robust geometric computing

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 of the two polytopes, while one of them 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 (i) to verify the correctness of the algorithm and its implementation, and (ii) 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 $P \oplus Q = \{p + q \mid p \in P, q \in Q\}$. Assume that a bounded polyhedron, referred to as polytope, R is moving in three-dimensional space. It is well known that R collides with another static polytope Q, if and only if the origin is in $-R \oplus Q$, where $-R$ 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., the Introduction in [1].

Minkowski sum is a common and practical operation with various applications in a wide variety of fields, such as computer graphics, motion planning in robotics, 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 for arbitrary polytopes. One common approach is to decompose each polytope into convex pieces, compute pairwise Minkowski sums of pieces of the two polytopes, and finally the union of the pairwise sums. Computing the Minkowski sum of two convex polytopes remains a key operation. This work deals with *convex* polytopes in $\mathbb{R}^3$, which are hereafter simply referred to as polytopes.

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 one step toward efficient and robust collision detection during such motion.

The collection of incidence relations between the vertices, edges, and facets of the Minkowski sum polytope, referred to as its combinatorial structure, remains the same when the summand polytopes translate. However, the structure can dramatically change when one, or both, of the polytopes rotate. (This is made clear by the introduction of Gaussian maps and their properties below.) Hence, one way to deal with rotations is to recalculate

[☆] This work has been supported in part by the Israel Science Foundation 236/06, by the German-Israeli Foundation 969/07, by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University, and by the EU Project under Contract No. 255827 (CGL—Computational Geometry Learning).

[*] Corresponding author.
E-mail addresses: naamamay@post.tau.ac.il (N. Mayer), efif@post.tau.ac.il (E. Fogel), danha@post.tau.ac.il (D. Halperin).

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.06.021

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 [2, Section 13.5] [3,4]. 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* Minkowski sums along the path (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 special, multi-precision number types. We use algebraic number types when rational numbers are insufficient, though reducing the performance impact by using an efficient type of algebraic numbers known as the sqrt-extension; see Section 2.5.

Computing the exact Minkowski sum in 3D is quite complex and non-essential for many applications. In some cases, computing an approximation of the boundary of the Minkowski sum, while providing a bound on the approximation, is sufficient. One example is the method suggested by Varadhan and Manocha [5], which results with an approximation of the 3D Minkowski sum of arbitrary polyhedral objects. It is based on the aforementioned decomposition approach, where the final union is approximated. They provide a two-sided Minkowski sum bound on the approximation. Ghosh [6] represents each input polygon or polyhedron with a slope diagram utilizing the concept of negative shapes and computes the Minkowski addition and decomposition of boundary-represented objects. Gritzmann and Sturmfels [7] present a polynomial-time algorithm for computing the Minkowski sum of k polytopes in $\mathbb{R}^d$.

Several output-sensitive methods were especially designed for exact construction of Minkowski sums of polytopes. Fogel and Halperin [1] represented every polytope as a Gaussian map and constructed their Minkowski sum by producing it from their overlay. Hachenberger [8] suggested to calculate Minkowski sums using Nef polyhedra, and Fukuda [9] provided a polynomial-time algorithm for variable number of polytopes and dimensions. This method was implemented by Weibel [10]. Our own work, which is known to achieve the best results (see comparisons in [1]) serves as the groundwork for the current results.

Less effort has been invested in efficient construction of Minkowski sums under rotation. A novel method was proposed recently by Lien [11]. Lien computes an initial Minkowski sum from scratch for the initial position of the two polytopes, and for every rotation of one of the polytopes he computes the new Minkowski sum structure by applying all the local structural changes caused by the rotation. 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 [12] for a survey of related problems and techniques. Among the vast number of publications in this area, we cite a small sample. The algorithm by Lin and Canny for collision detection and its optimized versions [13–16] maintain the shortest distance between two polytopes and update it according to their movement. The algorithm by Gilbert, Johnson, and Keerthi [17] and its enhancement [18] obtain the shortest distance between the polytopes by using simplices from both polytopes. The algorithm by Kim and Rossignac approximate collisions under screw motions [19].

We present a novel method for the fast retrieval of exact Minkowski sums of pairs of convex polytopes in $\mathbb{R}^3$ undergoing rotation. We investigate the space of combinatorially distinct sums when one of the polytopes undergoes rotation about one, two, or three axes. In our implementation and experiments we focus on the case of a rotation around a single arbitrary axis in space.

Ours is the first study that reveals *all* the combinatorially distinct Minkowski sums in the case of rotation, successfully coping with degeneracies and unrestricted by resolution parameters. In that we make a unique contribution to the expanding toolbox for handling Minkowski sums, paving the way to using Minkowski sums of polytopes that can change their orientation in applications that require high precision such as packing problems (for example trunk packing), object placement in tight quarters, and assembly planning. One may wonder whether following the exact geometric paradigm does not hamper performance in computationally intensive applications. However, with the fast progress in exact geometric computing machinery, and since we mostly rely on generic programming where one can easily upgrade auxiliary tools with not much programming effort, solutions as ours are becoming progressively more efficient.

A secondary contribution of our work is the careful and complete manipulation (construction and update) of the Gaussian map of the polytopes and of their Minkowski sum. The Gaussian map is a fundamental structure in geometric computing, and we believe that the intricate questions that arise in our work are inevitable, and will come up wherever Gaussian maps need to be updated. The intricacies arise since we need to update a subdivision on a sphere rather than in the plane, which turns out to be significantly more complex.

The rest of the paper is organized as follows. Section 2 reviews the spherical Gaussian map framework and the sqrt-extension number type, which we use in this work. Section 3 defines the Minkowski sum induced criticality maps in one, two, and three dimensions. Section 4 describes two enhancements to the original algorithm one for rotation about an arbitrary axis and another for economical preprocessing of the criticality map. The outcome of various experiments is reported in Section 5. Directions for future work are outlined in Section 6.

2. Preliminaries

In this section we introduce the extended Gaussian map—a unique dual representation of a polytope. An extended Gaussian map is in turn represented by an arrangement of geodesic arcs embedded on a sphere. This data structure is supported by the *2D Arrangements* package of CGAL [20]. The package also supports the overlay operation of such arrangements used to compute the Minkowski sum of polytopes represented by the arrangements; see [21–23] and Section 2.2 for more details.

CGAL in general, and the CGAL *2D Arrangements* package in particular, follow the exact geometric-computation (EGC) paradigm. EGC, as summarized by Yap [24], simply amounts to ensuring that we never err in predicate evaluations. EGC represents a significant relaxation from the naive concept of numerical exactness. Here, computation is carried out using a number type that supports only inexact arithmetic (e.g., double-precision floating point arithmetic), while analyzing the computation correctness. If the computation reaches a stage of uncertainty, the computation is re-done using unlimited precision. In cases