Exact Minkowski Sums and Applications*

Eyal Flato† Dan Halperin‡

Abstract

The *Minkowski sum* of two sets P and Q in $\mathbb{R}^d$ is the set $\{p+q \mid p \in P, q \in Q\}$. Minkowski sums are useful in robot motion planning, computer-aided design and manufacturing (CAD/CAM) and many other areas. We present a software package implemented at Tel Aviv University for the *exact* and efficient construction of Minkowski sums of planar sets. The package was implemented using the CGAL and LEDA software libraries.

1 Introduction

Given two sets P and Q in $\mathbb{R}^d$, their *Minkowski sum* (or vector sum), denoted by $P \oplus Q$, is the set $\{p+q \mid p \in P, q \in Q\}$. Minkowski sums are used in a wide range of applications, including robot motion planning [8], assembly planning [6], and computer-aided design and manufacturing (CAD/CAM) [2].

Consider for example an obstacle P and a robot Q that moves by translation. We can choose a reference point r rigidly attached to Q and suppose that Q is placed such that the reference point coincides with the origin. If we let Q' denote a copy of Q rotated by 180° degrees, then $P \oplus Q'$ is the locus of placements of the point r where $P \cap Q \neq \emptyset$. In the study of motion planning this sum is called a *configuration space obstacle* because Q collides with P when translated along a path π exactly when the point r, moved along π, intersects $P \oplus Q'$.

Much work has been done on obtaining sharp bounds on the size of the Minkowski sum of two sets in two and three dimensions, and on developing fast algorithms for computing Minkowski sums.

In this abstract we describe a software package, EMINK, developed at Tel Aviv University for the *exact* and efficient construction of Minkowski sums of planar sets—the package is briefly described in the next section. Beside robot motion planning, we can use Minkowski sums to solve other related problems like minimum distance separation (arising in cartography) and object placement (arising for example in manufacturing cell layout design). Details about these applications are given in [3]. Figure 1 displays a screenshot of the interactive program.

2 Algorithms and Implementation

We devised and implemented several algorithms for computing the Minkowski sum of two polygonal sets in $\mathbb{R}^2$ [3]. Our implementation is based on the CGAL software library (www.cgal.org). Our main goal was to produce a robust and exact implementation. This goal was achieved by employing

*This work has been supported in part by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces), by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Center for Geometric Computing and its Applications), and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

†Department of Computer Science, Tel Aviv University, Tel-Aviv 69978, Israel. flato@post.tau.ac.il.

‡Department of Computer Science, Tel Aviv University, Tel-Aviv 69978, Israel. danha@post.tau.ac.il.

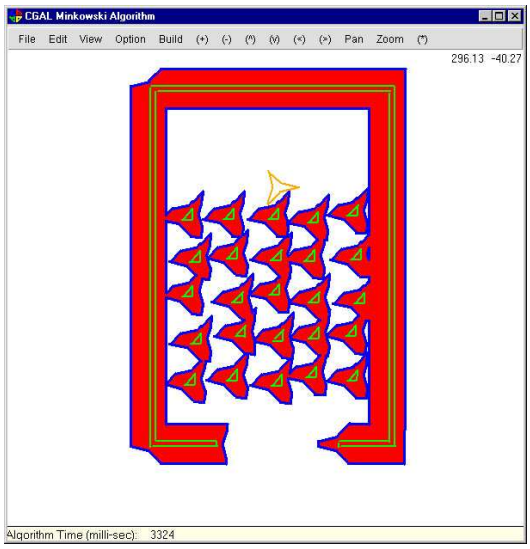

Figure 1: EMINK — screenshot of the interactive program

the CGAL *planar maps* and *arrangements* package [4], [7] while using exact number types. We use rational numbers and filtered geometric predicates from LEDA — the Library of Efficient Data-structures and Algorithms [9]. The connection between Minkowski sums and arrangements as well as related implementation projects are described in [5].

The planar maps and arrangements package of CGAL has several features that facilitated the implementation of EMINK. The planar subdivision that represent the Minkowski sum of two polygons often raises degenerate situations such as many segments incident to the same point or overlapping segments. The robust treatment of these cases, as supported by the package, is crucial for the correct construction of Minkowski sums. Another important advantage of CGAL is the ability to use different *traits* classes.[1] In addition to robustness we could have an efficient implementation by integrating LEDA's rational numbers and filtered geometric predicates into our software using a special traits class.

We are currently using our software to solve translational motion planning problems in the plane. We are able to compute collision-free paths even in environments cluttered with obstacles, where the robot could only reach a destination placement by moving through tight passages, practically moving in contact with the obstacle boundaries. See Figure 2 for an example. This is in contrast with most existing motion planning software for which tight or narrow passages constitute a significant hurdle.

The robustness and exactness of our implementation come at a cost: they slow down the running time of the algorithms in comparison with a more standard implementation that uses floating point arithmetic. This makes it especially necessary to try and speed up the algorithms in other ways. All our algorithms start with decomposing the input polygons into convex subpolygons. We discovered that not only the number of subpolygons in the decomposition of the input polygons but also their shapes had dramatic effect on the running time of the Minkowski-sum algorithms.

We implemented a dozen different methods for decomposing polygons and tested their suitability for efficient construction of Minkowski sums. We implemented various well-known decompositions as well as several new decomposition schemes which are all available in our package. In [1]

[1]Traits class — the geometric (not combinatorial) part of CGAL's arrangement package which includes the algebraic computations; see [4].

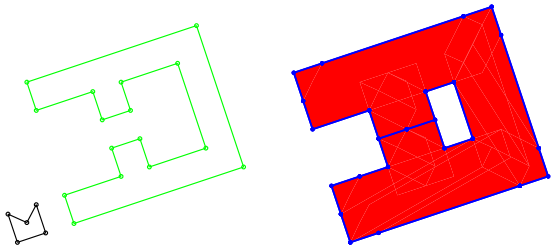

Figure 2: Tight passage: the desired target placement for the small polygon is inside the inner room defined by the larger polygon (left-hand side). In the configuration space (right-hand side) the only possible path to achieve this target passes through the line segment emanating into the hole in the sum.

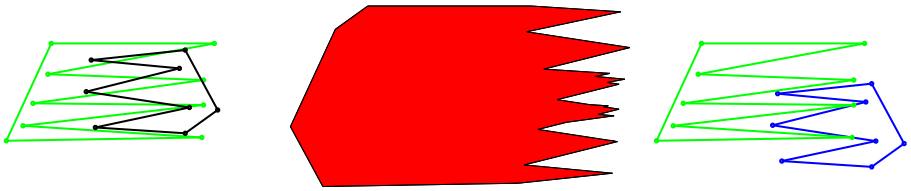

Figure 3: Minimal distance separation: the two polygons in their original placement (left-hand side), their Minkowski sum (middle) and after the minimal separation (right-hand side).

we report on our experiments with various decompositions and different input polygons. Among our findings are that in general: (i) triangulations are too costly (ii) what constitutes a good decomposition for one of the input polygons depends on the other input polygon — consequently, we develop a procedure for simultaneously decomposing the two polygons such that a "mixed" objective function is minimized, (iii) there are optimal decomposition algorithms that significantly expedite the Minkowski-sum computation, but the decomposition itself is expensive to compute — in such cases simple heuristics that approximate the optimal decomposition perform very well.

Note that convex *covering* (rather than decomposition) is also suitable as a first step in the construction of Minkowski sums. In our work however we focused on convex decomposition.

Recently the package has been extended to construct Minkowski sums of polygons and discs.[2] Since in this case the coordinates of the boundary of the sums need no longer be rational this requires the use of a more elaborate number type: LEDA's "real" [9].

3 Applications

The translational robot motion planning problem is a fundamental case study for Minkowski sum algorithms, but there are many more applications in which the Minkowski sum operation is a useful tool. Two examples which we solve within EMINK are listed here.

Given two polygons find the minimum length translation of one polygon relative to the other that will make the two polygons interior disjoint; see Figure 3. Assuming that in their original placement P and Q intersect and that the reference point of Q is at the origin O, it is not difficult to see that the minimum translation of Q relative to P is described by the point on the boundary of $P \oplus Q'$ which is closest to O where Q' is a copy of Q rotated 180°.

Given two polygons P and Q in the plane, another widely studied problem is to find whether P can be fully placed inside Q. This problem is known as the polygon containment problem. If we restrict it to a translational problem (namely the orientation of P is fixed) it can be solved as

[2]The extension to discs is by Eran Leiserowitz and is based on recently developed tools due to Shai Hirsch and Ron Wein.

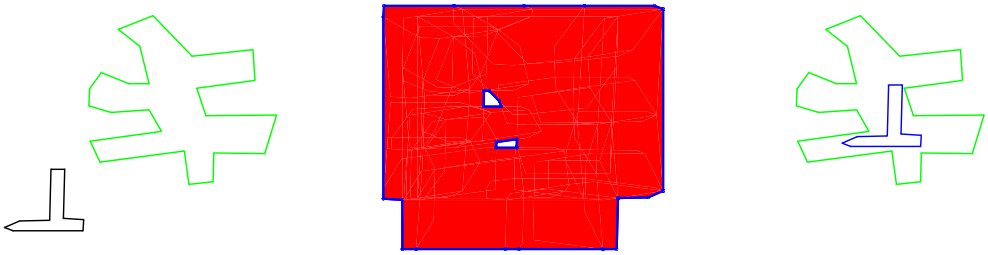

Figure 4: Polygon containment: the input polygons P and Q are displayed on the left-hand side, $P' \oplus \overline{Q}$ is in the middle, and a possible placement for P inside Q is on the right-hand side.

follows: consider the complement of Q as an obstacle for the robot P and seek a placement of P such that it does not penetrate the obstacle. Practically, let B be the bounding box of Q and let $\overline{Q} = B \setminus Q$. The free placements for P inside Q can be found by computing $P' \oplus \overline{Q}$ were P' is P rotated by 180°. See Figure 4 for an example.

4 Acknowledgements

We wish to thank Shai Hirsch, Eti Ezra, Eran Leiserowitz, and Efi Fogel for their valuable contribution to this work. We also thank all others who participate in developing CGAL and especially CGAL members in Tel-Aviv University.

References

[1] P. K. Agarwal, E. Flato, and D. Halperin. Polygon decomposition for efficient construction of Minkowski sums. *Comput. Geom. Theory Appl.*, 21:39–61, 2002.

[2] G. Elber and M.-S. Kim, editors. *Special Issue of Computer Aided Design: Offsets, Sweeps and Minkowski Sums*, volume 31. 1999.

[3] E. Flato. Robust and efficient construction of planar Minkowski sums. Master's thesis, Dept. Comput. Sci., Tel-Aviv Univ., 2000. http://www.cs.tau.ac.il/~flato.

[4] E. Flato, D. Halperin, I. Hanniel, O. Nechushtan, and E. Ezra. The design and implementation of planar maps in CGAL. *The ACM Journal of Experimental Algorithmics*, 5, 2000. Also in LNCS Vol. 1668 (WAE '99), Springer, pp. 154–168.

[5] D. Halperin. Robust geometric computing in motion. In B. Donald, K. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Dimensions (WAFR '00)*, pages 9–22. A. K. Peters, Wellesley, MA, 2001. To appear in *International Journal of Robotics Research.*

[6] D. Halperin, J.-C. Latombe, and R. H. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 26:577–601, 2000.

[7] I. Hanniel and D. Halperin. Two-dimensional arrangements in CGAL and adaptive point location for parametric curves. In *Proc. of the 4th Workshop of Algorithm Engineering*, volume 1982 of *Lecture Notes Comput. Sci.*, pages 171–182. Springer-Verlag, 2000.

[8] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[9] K. Melhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, 1999.