Fast and Robust 2D Minkowski Sum Using Reduced Convolution

Evan Behar

Jyh-Ming Lien

Abstract—We propose a new method for computing the 2-d Minkowski sum of non-convex polygons. Our method is convolution based. The main idea is to use the *reduced convolution* and filter the boundary by using the topological properties of the Minkowski sum. The main benefit of this proposed approach is from the fact that, in most cases, the complexity of the complete convolution is much higher than the complexity of the final Minkowski sum boundary. Therefore, the traditional approach often wastes a large portion of the computation on computing the arrangement induced by the complete convolution that is later on thrown away. Our method is designed to specifically avoid this waste of computation. We experimentally demonstrate that the proposed method is more efficient than the existing methods.

I. INTRODUCTION

The Minkowski sum (M-sum) plays important roles in many sub-fields of robotics. For example, it is the basic operation for mapping the workspace to the configuration space, and is fundamental in many geometric applications such as estimating the penetration depth between two intersecting objects. The M-sum of two shapes P and Q is defined as:

$$P \oplus Q = \{p + q \mid p \in P, q \in Q\}. \quad (1)$$

The idea of using M-sums to solve these problems has been proposed for more than three decades, and many methods have been proposed to compute M-sums. However more efficient and robust methods have only been proposed recently. Surveys can be found in [1], [2], [3]. In particular for *convex shapes*, several efficient methods [4], [3], [5], [6] are known to compute the M-sum in both 2-d and 3-d.

For computing the M-sum of non-convex shapes, many methods are based on the idea of convex decomposition because computing the M-sum of convex shapes is easier than non-convex shapes. In this framework, the pairwise M-sums of the convex components are computed and, then all these pairwise M-sums are united to form the final M-sum. Although conceptually simple, this approach is usually not practical due to the size of the decomposition and the difficulty in implementing a robust union operation.

Convolution-based methods do not have these problems. The convolution of two shapes P and Q , denoted as $P \otimes Q$, is a set of line segments in 2-d or facets in 3-d. The segments or facets of the convolution are generated by “combining” the segments or the facets of P and Q [7]. One can think of the convolution as the M-sum that involves only the boundary, i.e., $P \otimes Q = \partial P \oplus \partial Q$. It is known that the convolution

forms a superset of the M-sum [1], i.e., $\partial(P \oplus Q) \subset P \otimes Q$. If both P and Q are convex, $\partial(P \oplus Q) = P \otimes Q$. Otherwise, it is necessary to trim the convolution to obtain the M-sum boundary.

One of the main limitation of the convolution-based methods comes from the mismatch between complexity of the arrangement induced by the convolution and the complexity of the Minkowski sum. As we will show in the experimental results, the complexity of the Minkowski sum is in most cases significantly lower than that of the convolution and its arrangement. Computing the arrangement induced by the convolution is not only expensive but also wasteful because many elements in this arrangement are later on thrown away. Motivated by this observation, an obvious approach to improve the efficiency is to avoid computing the complete convolution and its arrangement. However, by computing only a subset of the convolution, the main technical challenge now becomes how we can define a smaller set of the convolution while still being able to extract the M-sum.

A. Our Contribution

We propose an improved method for computing the 2-d Minkowski sum (M-sum) of non-convex polygons using the idea of *reduced convolution* (defined later in Section IV). We then extract the boundaries by filtering unwanted geometries from the reduced convolution using the topological properties of the M-sum. A detailed description of the proposed method can be found in Section IV; an example output from our method can be found in Fig. 1.

As we will show in the end of this paper, our method is easy to implement and provides better efficiency than the state of art. The proposed method can handle certain type of degenerate cases, such as non-simple polygons and can be easily modified to perform morphological operations, such as erosion, directly on the polygons. Our implementation is available to the public and can be download from our project webpage [8].

II. RELATED WORK

During the last three decades, many methods have been proposed to compute the Minkowski sums (M-sum) of polygons or polyhedra. Most of these methods can be categorized into one of the two main frameworks: divide-and-conquer and convolution. Here we briefly review these techniques. A more detailed review can be found in our previous work [9].

Divide-and-Conquer. This approach is first proposed by Lozano-Pérez [10] to compute $\mathcal{C}$ -obst for motion planning. Although the main idea of this approach is simple, the divide step (i.e., convex decomposition) and the merge step (i.e.,

This work is supported in part by NSF IIS-096053.

Both authors are with Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030 USA, {ebehar, jmlien}@cs.gmu.edu

Fig. 1. An example output of the proposed method. The entire computation takes 0.3 seconds on a regular desktop. (a) Input polygon with 18 holes and 1,815 vertices. (b) The reduced convolution of (a) and a circle with 32 vertices. (c) The M-sum of (a) and a circle.

union) can be very difficult to implement robustly in practice, in particular when the input shapes are complex, e.g., non-manifold or self-intersecting polygons. In addition, Agarwal et al. [11] have shown that different decomposition strategies can greatly affect the efficiency of this approach. Moreover, they show that the efficiency of any particular decomposition technique depends greatly on the input polygons.

The union step is even more troublesome. This is in particular true because the size and the complexity of the geometry generated during the intermediate steps can be overwhelming. Two polygons decomposed into only 100 components each can have 10K pairwise M-sums. While methods exist to perform the union operation, no existing method can both efficiently and robustly compute the union of tens of thousands of pairwise M-sums. A recent attempt to gain better efficiency is by Flato [12], who computes the unions using a strategy that combines arrangement with incremental insertion.

Convolution. For 2-d polygons, Guibas and Seidel [13] show an output sensitive method to compute convolution curves. Later, Ghosh [1] proposed an approach, which unifies 2-d and 3-d, convex and non-convex, and M-sum and decomposition operations. The main idea in his method is the negative shape and slope diagram. The slope diagram is closely related to the *Gaussian map*, which has been recently used by Fogel and Halperin [3] to implement robust and efficient Minkowski sum computation of convex objects.

The main difficulty of the convolution-based methods is to remove the portion of the facets that are inside the M-sum. Barki et al. [14], [15] demonstrate an exact convolution-based method in 2-d predicated on recovering the boundary from a set of orthographic projections in $\mathbb{R}^3$. The primary limitations of the method proposed in [14], [15] are that it is limited to (1) at most one non-convex summand; (2) that non-convex polygon cannot contain a fold; and (3) the method proposed does not handle polygons with holes.

Ramkumar [16] uses a convolution-based method that detects self-intersections using a ray-shooting method and removes them, leaving only the M-sum boundary. Ramkumar's algorithm is asymptotically slower than existing algorithms of its time though its practical performance is acceptable. It is however overshadowed in performance by more recent convolution methods.

Fig. 2. Two polygons P and Q shown with the edges' outward normals.

Recently, Wein [17] showed a robust and exact method for non-convex polygons. To obtain the M-sum boundary from the convolution, his method computes the arrangement induced by the line segments of the convolution and keeps the cells with non-zero winding numbers. Wein's method consistently outperforms the best decomposition methods except on polygons where the geometry can be easily decomposed into pairwise interior disjoint pieces.

Farouki and Hass [18] demonstrate a method for computing the M-sum boundary of general closed curves. They compute characteristic points on the envelope curve of the convolution, and use these characteristic points to avoid many expensive geometric operations necessary to trim from the interior of convolution—however these operations are only useful on curves and in fact the method described is more expensive on polygons. Indeed, they admit that if one can approximate the curves sufficiently with polygons, one could use existing polygon methods to compute an approximate M-sum sum more efficiently. From this we can conclude that this method is not suitable for computing the M-sum when we are guaranteed that the inputs are polygons.

III. PRELIMINARIES

In this section, we define the notations that will be used throughout the paper. Let P and Q be simple polygons composed of n and m (counterclockwise) ordered vertices, respectively. We denote the vertices of P as $\{p_i\}$ and the edge that starts at vertex p_i as $e_i = \overline{p_i p_{i+1}}$. The edge e_i has two associated vectors, the vector from p_i to p_{i+1} , i.e., $\vec{v}_i = \overline{p_i p_{i+1}}$, and the outward normal $\vec{n}_i$. The definition for the vertices $\{q_j\}$ and edges of Q is the same. Fig. 2 shows an example of P and Q .

The boundary of the Minkowski sum (M-sum) of P and Q is known to have complexity $O(m^2n^2)$ [19] and is composed of an external boundary and a (possibly empty) list of hole boundaries. Our approach for computing the M-sum boundary is based on the idea of convolution. Specifically, an edge $\overline{p_i p_{i+1}}$ of P and a vertex q_j of Q (or vice versa) form a segment of $P \otimes Q$ if $\overrightarrow{p_i p_{i+1}} \in [q_{j-1} q_j, q_j q_{j+1})$, and we say that $\overline{p_i p_{i+1}}$ and q_j are *compatible*. Equivalently, $\overline{p_i p_{i+1}}$ and q_j are compatible if the outward normal of $\overline{p_i p_{i+1}}$ lies between the normals of the incident edges of q_j . For example, in Fig. 2, $\overline{p_3 p_1}$ is compatible vertices q_2 and q_3 .

IV. OUR METHOD

We propose a new method to compute the M-sum of non-convex polygons. Similar to Wein's method [17], our method can be considered as a type of convolution-based approach. However, unlike Wein [17], the proposed method avoids computing (1) the complete convolution, (2) the arrangement of the segments of the convolution, and (3) the winding number for each arrangement cell. Our method is sketched in Algorithm IV.1.

Algorithm IV.1: M-SUM(P, Q)

R = Compute the reduced convolution of P and Q
 L = Extract orientable loops from R
 M = Filter boundaries from L
return (M)

Algorithm IV.1 first computes a subset of the segments that is from the convolution of the inputs. We call this subset a “reduced convolution.”

Definition 4.1: A **reduced convolution** is a set of segments $\overline{p_i p_{i+1}} \oplus q_j$ and $p_k \oplus \overline{q_l q_{l+1}}$ and q_j and p_k must be convex.

Then Algorithm IV.1 identifies closed loops that are (1) non-overlapping and (2) *orientable*. We say that a loop is orientable if all edges in the loop have normals pointing consistently inward or consistently outward. These loops form potential boundaries of the M-sum and are further filtered by analyzing their nesting relationship. Finally, the remaining boundaries are filtered by checking the intersections between the input polygons placed at the configurations along these loops. Each of these steps is discussed in detail below.

A. Reduced convolution

In the first step of the algorithm, we compute a subset of the segments of the convolution based on the following observation.

Observation 4.2: Given a convolution segment $s = e_i \oplus q$ of an edge $e_i \in P$ and a vertex $q \in Q$, if q is a reflex vertex, s must not be a boundary of the M-sum of P and Q . This observation remains true if $s = p \oplus e_j$, where $p \in P$ is a reflex vertex and $e_j \in Q$ is an edge.

The proof is omitted due to space limitations, for details please see our technical report [20].

Because of the definition of a reflex angle, the number of edges that are compatible with any convex vertex in Q

Fig. 3. Steps for computing the M-sum of two simple polygons. In (a), a star and a slightly rotated copy of it are shown.

form a lower bound on the number of edges compatible with any reflex vertex in Q . Due to this, the number of segments filtered by Observation 4.2 is significant. An example of this is demonstrated in Fig. 3. Figs. 3(a) and 3(b) show the input polygons and the full convolution of the inputs, respectively. Fig. 3(c) demonstrates the reduced convolution of the same input polygons, in which there are significantly fewer convolution edges.

B. Orientable Loop Extraction

Now, since the segments that we will be working with are no longer a complete convolution, we cannot apply the idea of computing the winding number for each arrangement cell to extract the M-sum boundary as done in [17]. Instead, we proceed by defining two filters.

Observation 4.3: We observe that the boundary of the Minkowski sum must be an orientable loop (if it encloses an area, either positive or negative).

We say that a loop is orientable if all the normal directions of the edges in the loop are all either pointing inward or outward. Note that the segments we considered are edges from P and Q , therefore, they are directional (as vertices in P and Q are ordered) and include normal directions pointing outward (to P or Q). Therefore, given two adjacent segments

$s = \{u, v\}$ and $s' = \{v, u'\}$ sharing an end point v , we can check whether s and s' belong to an orientable loop if

$$\overrightarrow{uv} \times \overrightarrow{n_s} = \overrightarrow{vu'} \times \overrightarrow{n_{s'}}, \quad (2)$$

where $\overrightarrow{n_x}$ is the normal vector of segment x , and $\times$ is the cross product. If s and s' satisfy Eq. 2, we say they are compatible segments.

To extract all orientable loops, we compute the intersections of the segments and split all segments at the intersections. A loop is then traced by starting at an arbitrary segment s that has not been considered and then iteratively including compatible segments adjacent to s . Note that there can be multiple compatible segments adjacent to s and all are incident to a single point v . This problem is in fact easy to handle since all M-sum boundaries must be manifold. Thus, we simply pick the segment that makes the *largest clockwise turn* from s among all the incident segments.

Observation 4.4: The loops must obey the nesting property, i.e., the loops that are *directly* enclosed by the external loop must be holes and will have negative areas, and the loops that are directly enclosed by the holes must have positive areas. The polygon in Fig. 3(d) has an improperly nested loop (positive area inside positive area) that must be removed; once the loop is removed, we get the correct M-sum as seen in Fig. 3(e).

This is because all loops we generated are non-overlapping (i.e., they don't intersect or touch) due to the manifold properties. The nesting property can be determined efficiently using a plane sweep algorithm, e.g., [21], in $O(n \log n)$ time for n segments.

C. Boundary Filtering

So far, we have introduced three quite efficient filters based on Observations 1 through 3. Unfortunately, some of the remaining loops may not be boundaries of the M-sum. An example of the false loop is the hole boundary shown in Fig. 4(a). These false hole boundaries are the direct result from the filters of Observations 1 through 3 that exploit only local topological properties on the convolution.

Therefore, we will have to resort to collision detection, a global operation, to remove all the false loops. Given a translational robot P and obstacles Q , the contact space of P and Q can be represented as $\partial((-P) \oplus Q)$, where $-P = \{-p \mid p \in P\}$. If a point x is on the boundary of the M-sum of two polygons P and Q , then the following condition must be true:

$$(-P^\circ + x) \cap Q^\circ = \emptyset,$$

where Q° is the open set of Q and $(P+x)$ denotes translating P to x . Figs. 4(b) to 4(e) show that the false hole boundary provides collision-free configurations only to the body of the airplane but collides the tail with the obstacle.

Although there are many methods to optimize the computation time for collision detection, collision detection is more time consuming than the previous filters. Fortunately, only a single collision detection is needed to reject or accept a loop based on the following lemma.

Lemma 4.5: All the points on a false hole loop must make P collide with Q .

Again, we must omit the proof due to space limitations; our technical report contains the proof. [20]

V. EXPERIMENTAL RESULTS

In this section, we show that the computation time of the proposed method is more efficient than the traditional approach. All the experiments are performed on a machine with Intel CPUs at 2.13 GHz with 4 GB RAM. Our implementation is coded in C++. For accurate floating-point computations, we use *GNU MPFR* [22] and its wrapper *MPFRC++*.

In our current implementation, the line segment intersection and the collision detection between P and Q are performed by exhaustively checking all pairs of segments. Fig. 5 shows 11 models that we will use in the experiments.

A. Computation Time

We first compare the computation time of the proposed method to the convolution implementation provided by CGAL [23]. As stated in the CGAL documentation, the CGAL code implements Wein's idea [17]. We use all the models in Fig. 5 in this experiment and compute the M-sums from all pairs. The results are shown in Fig. 6.

In all examples, our method is faster than CGAL. The smallest speedup is 1.01 from *grate3* $\oplus$ *monkey*, and the largest speedup is 30.8 from *bird* $\oplus$ *hand1*. Examination of Figs. 6 and 7(b) suggests that the ratio of intersections in the reduced convolution to the intersections in the full convolution is not a good predictor of speedup. Another interesting thing to note, by comparing Figs. 6 and 7(a) is that the percentage of time spent in collision detection is also not a good predictor of speed-up.

Fig. 6. Speedup over CGAL implementation. Large black dots are the results involving interesting model pairs: (A) *G3*+*monkey* [smallest speedup] (B) *monkey* $\oplus$ *monkey* [most complex convolution] is not shown in the speedup graph because CGAL did not terminate on this input pair (C) *G1* $\oplus$ *G2* [largest percent computation time performing collision detection] (D) *hand1* $\oplus$ *hand2* [least complex convolution] (E) *bird1* $\oplus$ *hand1* [largest speedup]. Other dots are from the rest of the models. Each dot is computed as t_{CGAL}/t_{ours} , where t_{CGAL} and t_{ours} are the computation times for a specific pair of models using CGAL and our method, respectively.

G1 $\oplus$ *G2* (label C in the graphs), which spends 26.9% of its run-time in collision detection takes only 14.26ms to complete, 3.34ms of which is spent in collision detection.

Fig. 4. (a) The hole boundary is a false loop. (b-e) The configurations in the false loop make the airplane collide with the obstacle. Notice that the collisions only occur around the tail of the airplane.

Fig. 5. Models used in the experiments. The figure shows the names and the sizes of the polygons. Note that (c) is G3 (24), and (i) is hand2 (84).

Next, we look further into the behavior of the proposed method. We break down the computation time into four parts: (1) the time for computing the reduced convolution, (2) time for computing the intersections of the segments in the reduced convolution, (3) time for identifying the non-intersecting orientable loops, and (4) time for filtering hole boundaries. In all figures, the x axis indicates only an index assigned to the experiment. In this experiment, we use only the larger models in Fig. 5 (airplane, bird, dog, G1, G2, G3, G4, hand1, hand2, mammoth and monkey). The results obtained from all pairs of these larger models are shown in Fig. 7(a).

One may be concerned that the use of collision detection for filtering out the remaining holes will be the bottleneck of the entire computation. Fig. 7(a) shows that this is in fact not the case. In all of the examples we have tested, the filtering step only takes less than 40% of the time. The bottleneck is in fact in finding the line segment intersections of the convolution. Recall that in our implementation both collision detection and line segment intersection are calculated by checking line segments exhaustively between all pairs. Therefore, our current implementation still has room for much improvement.

B. Reduced Convolution vs. Complete Convolution

In this experiment, we show that the size of the reduced convolution is the key that the proposed method is more efficient. Theoretically, the reduced convolution is at most half of the complete convolution. Practically, this assumption may be off. Fig. 7(b) shows exactly this. Again, we use the models in Fig. 5 and compute the M-sums of all pairs. We study the differences when the reduced convolution and the complete convolution are used.

Fig. 7(b) clearly shows that the size of the arrangement of the reduced convolution is significantly smaller than that of the regular convolution. Note that the y axis in this plot is in logarithmic scale. In the best case (grate4 $\oplus$ monkey), the reduced convolution is 314.7 times smaller. In this case, the reduced convolution has 29,411 intersections and the full convolution has 9,255,477 intersections. In the worst case (G3 $\oplus$ G3), the reduced convolution is only 1.4 times smaller. In this case, the reduced convolution has 774 intersections and the full convolution has 1,087 intersections.

VI. CONCLUSION

In this paper, we propose a new method based on the idea of the *reduced convolution* for computing the 2-d Minkowski sum of non-convex polygons. We proposed several filters in order to extract the boundaries of the Minkowski sum

Fig. 7. (a) Normalized computation time for each step in our algorithm. The total computation for each M-sum is normalized to 1. The black part is the time spent computing the reduced convolution itself. The dark grey part indicates the time for computing the intersections of the segments in the reduced convolution. The grey part is the time for identifying the loops. The light grey is the time for filtering hole boundaries. (b) Number of segment intersections in the reduced convolution and the complete convolution. The arrangement of the reduced convolution is significantly smaller than that of the regular convolution. Note that the y axis is in logarithmic scale.

using the topological properties of the Minkowski sum. The proposed method has been implemented and extensively tested. Our experimental results in Section V-A showed that the proposed method is more efficient. In Section V-B, we further showed that the efficiency gain of this proposed approach is indeed from the fact that, in all cases studied, the complexity of the complete convolution is much higher than the complexity of the reduced convolution and that of the final Minkowski sum boundary.

In the near future, we plan to further improve the efficiency of the proposed method by using better collision detection and segments intersection methods. For the long term goal, the result of this work will be used in providing a more efficient way to compute the C -space mapping. Our preliminary result [20] shows that, using the approach developed in this paper, C -space mapping can be made simpler in implementation and often more efficient than the existing techniques.

REFERENCES

- [1] P. K. Ghosh, "A unified computational framework for Minkowski operations," *Computers and Graphics*, vol. 17, no. 4, pp. 357–378, 1993.
- [2] G. Varadhan and D. Manocha, "Accurate Minkowski sum approximation of polyhedral models," *Graph. Models*, vol. 68, no. 4, pp. 343–355, 2006.

- [3] E. Fogel and D. Halperin, "Exact and efficient construction of Minkowski sums of convex polyhedra with applications," in *Proc. 8th Workshop. Alg. Eng. Exper. (Alenex'06)*, 2006, pp. 3–15.
- [4] A. Kaul and J. Rossignac, "Solid-interpolating deformations: construction and animation of PIPs," in *Proc. Eurographics '91*, 1991, pp. 493–505.
- [5] P. Gritzmann and B. Sturmfels, "Minkowski addition of polytopes: computational complexity and applications to Gröbner bases," *SIAM J. Discret. Math.*, vol. 6, no. 2, pp. 246–269, 1993.
- [6] K. Fukuda, "From the zonotope construction to the minkowski addition of convex polytopes," *Journal of Symbolic Computation*, vol. 38, no. 4, pp. 1261–1272, 2004.
- [7] L. J. Guibas, L. Ramshaw, and J. Stolfi, "A kinetic framework for computational geometry," in *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, 1983, pp. 100–111.
- [8] E. Behar and J.-M. Lien, "Project Webpage for Fast and Robust 2D Minkowski Sum Using Reduced Convolution," <http://masc.cs.gmu.edu/wiki/ReducedConvolution/>, 2011.
- [9] J.-M. Lien, "A simple method for computing Minkowski sum boundary in 3d using collision detection," in *The Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Guanajuato, Mexico, Dec 2008.
- [10] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, 1983.
- [11] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," in *European Symposium on Algorithms*, 2000, pp. 20–31.
- [12] E. Flato, "Robust and efficient construction of planar minkowski sums," M.Sc. Thesis, Dept. Comput. Sci., Tel-Aviv Univ., Israel, 2000.
- [13] L. J. Guibas and R. Seidel, "Computing convolutions by reciprocal search," *Discrete Comput. Geom.*, vol. 2, pp. 175–193, 1987.
- [14] H. Barki, F. Denis, and F. Dupont, "Contributing vertices-based minkowski sum computation of convex polyhedra," *Comput. Aided Des.*, vol. 41, no. 7, pp. 525–538, 2009.
- [15] —, "Contributing vertices-based minkowski sum of a non-convex polyhedron without fold and a convex polyhedron," in *Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on*, June 2009, pp. 73–80.
- [16] G. D. Ramkumar, "An algorithm to compute the minkowski sum outerface of two simple polygons," in *Polygons, Proc. ACM Symposium on Computational Geometry*. Press, 1996, pp. 234–241.
- [17] R. Wein, "Exact and efficient construction of planar Minkowski sums using the convolution method," in *Proc. 14th Annual European Symposium on Algorithms*, 2006, pp. 829–840.
- [18] R. T. Farouki and J. Hass, "Evaluating the boundary and covering degree of planar minkowski sums and other geometrical convolutions," *J. Comput. Appl. Math.*, vol. 209, pp. 246–266, December 2007. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1287834.1287914>
- [19] D. Halperin, "Robust geometric computing in motion," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 219–232, 2002.
- [20] E. Behar and J.-M. Lien, "A new method for mapping the configuration space obstacles of polygons," George Mason University, Tech. Rep. GMU-CS-TR-2010-11, 2010.
- [21] C. Bajaj and T. K. Dey, "Polygon nesting and robustness," *Inform. Process. Lett.*, vol. 35, pp. 23–32, 1990.
- [22] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A multiple-precision binary floating-point library with correct rounding," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 2, p. 13, 2007.
- [23] A. Fabri, G. Giezeman, L. Kettner, S. Schirra, and S. Schönherr, "On the design of CGAL a computational geometry algorithms library," *Software Practice and Experience*, vol. 30, no. 11, pp. 1167–1202, 2000.