Title:

On Computing Offsets of Polygons

Authors:
Martin Held, held@cs.sbg.ac.at, Universität Salzburg
Stefan de Lorenzo, slorenzo@cs.sbg.ac.at, Universität Salzburg
Peter Palfrader, peter@palfrader.org, Universität Salzburg

Keywords:
Offsetting, Boolean operations, Voronoi diagram, Bentley-Ottmann algorithm, Experimental evaluation

DOI: 10.14733/cadconfP.2023.174-178

Introduction:

For a set S of (possibly infinitely many) points in the plane, the *constant-radius offset* $\mathcal{O}(S, r)$ for offset distance r is the set of all points of the plane whose minimum (Euclidean) distance from S equals r. Mathematically speaking, such an offset is the envelope of the offset area $\mathcal{OA}(S, r) := \bigcup_{p \in S} D(p, r)$, where $D(p, r)$ denotes a disk of radius r centered at the point p. If S is given by a set of straight-line segments and circular arcs, then such an offset consists of one or more closed curves consisting of straight-line segments and circular arcs. See Figure 1a for a sample offset of a simple polygon. We call a closed curve formed by a (finite) sequence of straight-line segments and circular arcs a *circular-arc polygon*.

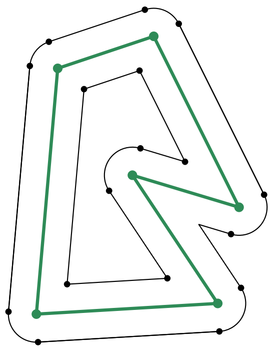
Held et al. [4] generalize the concept of multiplicatively weighted Voronoi diagrams (MWVDs) by introducing so-called *variably-weighted straight-line segments*: They assign real-valued weights $w(p) > 0$ and $w(q) > 0$ to the end points p and q of a straight-line segment $\overline{pq}$. Weights of the points along $\overline{pq}$ can be derived by linearly interpolating between $w(p)$ and $w(q)$. Even for a fixed offset value, these weights introduce more flexibility: In contrast to conventional constant-radius offsetting, where all parts of the input set expand or shrink uniformly at the same speed, *variable-radius offsetting* induced by the weights allows parts to expand or shrink in a non-uniform manner; see Figure 1b.

By definition, the offset area $\mathcal{OA}(S, r)$ is a union of disks. As shown in the sequel, if S is modeled by a finite number of points and straight-line segments, then such a union of possibly infinitely many disks can be replaced by the union of finitely many disks and rectangles, or by disks and trapezoids in the case of variably-weighted sites. Hence, constant-radius and variable-radius offset areas can be obtained by computing the union of a finite number of circular-arc polygons of low combinatorial complexity.

If no weights are assigned to the end points of the straight-line segments of S, then there is another alternative: The approach detailed by Held [3] employs the Voronoi diagram of S to obtain constant-radius offsets of S. We note that Voronoi diagrams can be generalized appropriately to structures that would admit efficient variable-radius offsetting even for weighted segments once that structure is known, see [4]. Unfortunately, these structures are much more difficult to compute than standard Voronoi diagrams.

Our Contribution:

We present an experimental evaluation of run-times consumed by the computation of constant-radius and variable-radius offsets of polygons. Our evaluation involves several leading software packages for



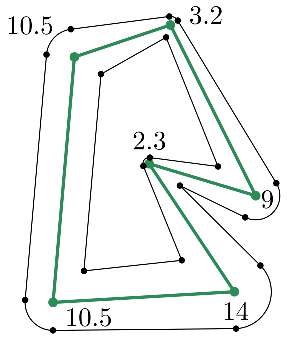


Fig. 1: Sample interior and exterior (a) constant-radius and (b) variable-radius offsets that have been derived from the same input polygon. The weights are written next to the corresponding polygon vertices.

computing Boolean operations on (circular-arc) polygons. Their run-times are compared to the results of our own codes Circular Arc Polygon OPerations (CAPOP) for computing Boolean operations and VRONI (for computing Voronoi diagrams). CAPOP can handle genuine circular-arc polygons without prior (or code-internal) polygonal approximations of circular arcs. Extensive tests show that CAPOP is slower than VRONI but that its performance is clearly superior compared to the other software packages that we tested. However, in contrast to VRONI, CAPOP can also generate variable-radius offsets.

Variable-Radius Offsets:

Let p be a point of the Euclidean plane $\mathbb{R}^2$ and assign a positive real-valued weight $w(p)$ to it. We call p a weighted point. The weighted distance $d_w(a,p)$ between a point $a \in \mathbb{R}^2$ and p is defined as $d_w(a,p) := \frac{d(a,p)}{w(p)}$, where $d(a,p)$ denotes the standard Euclidean distance between a and p. Now consider a straight-line segment $\overline{pq}$ between the weighted points p and q. The weights $w(p)$ and $w(q)$ need not be identical. For $0 \le \lambda \le 1$, the weight of a point $(1-\lambda)p + \lambda q$ on $\overline{pq}$ is given by $(1-\lambda)w(p) + \lambda w(q)$, i.e., by the matching linear interpolation of the weights of p and q. We call $\overline{pq}$ a *variably-weighted straight-line segment*. The weighted distance between a point $a \in \mathbb{R}^2$ and $\overline{pq}$ is given as $d_w(a, \overline{pq}) := \min_{b \in \overline{pq}} d_w(a,b)$.

Let S be a set of finitely many weighted points and variably-weighted straight-line segments defined by some pairs of the weighted points. No straight-line segment of S is allowed to contain any weighted point of S except for its two end points, and no pair of segments of S may share a point except for a common end point. We refer to the elements of such a set S as *sites*.

The variable-radius offset area $\mathcal{OA}_v(S,r)$ induced by S relative to the offset value $r \ge 0$ is the set of all points of the plane whose minimum weighted distance from S is at most r. More formally, $\mathcal{OA}_v(S,r) := \left\{a \in \mathbb{R}^2 : \min_{s \in S} d_w(a,s) \le r\right\}$. The variable-radius offset $\mathcal{O}_v(S,r)$ of S for offset value r is given by the boundary of its offset area. All variable-radius offsets are exclusively made up of straight-line segments and circular arcs [4].

The definition of $\mathcal{OA}_v(S,r)$ implies for a point a that $a \in \mathcal{OA}_v(S,r)$ if and only if there exists a site $s \in S$ such that $d_w(a,s) \le r$. That is, if and only if $a \in \mathcal{OA}_v(\{s\},r)$. If s is a weighted point, then $\mathcal{OA}_v(\{s\},r)$ is given by the disk $D(s, w(s) \cdot r)$. If s is a variably-weighted straight-line segment $\overline{pq}$, then $\mathcal{OA}_v(\{s\},r)$ is given by the convex hull of $\mathcal{OA}_v(\{p\},r)$ and $\mathcal{OA}_v(\{q\},r)$, i.e., by the convex hull of two disks. Hence, in this case $\mathcal{OA}_v(\{s\},r)$ is the union of these two disks and a trapezoid defined by the four points in which the two bi-tangents touch the disks. In Figure 2b, the offset of a variably-weighted straight-line segment is shown where the weight of q is twice as large as the weight of p.

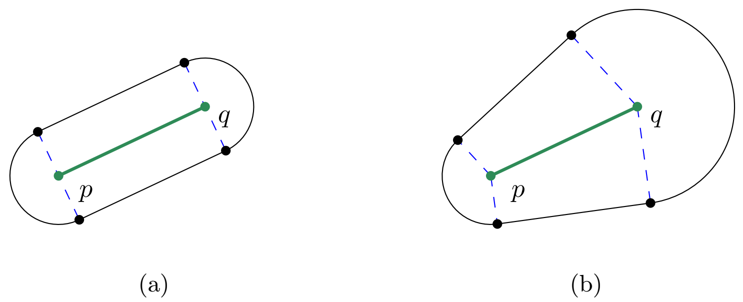


Fig. 2: (a) A constant-radius offset and (b) a variable-radius offset of a straight-line segment $\overline{pq}$ is shown.

Boolean Operations on Circular-Arc Polygons:

The previous section allows us to conclude that we can compute $\mathcal{OA}_v(S, r)$ by computing the union of the finitely many offset areas $\mathcal{OA}_v(\{s\}, r)$ for all $s \in S$. Since every $\mathcal{OA}_v(\{s\}, r)$ has a simple circular-arc polygon as its boundary, computing the union of circular-arc polygons would suffice to generate a variable-radius offset. This insight lets us resort to computing Boolean operations for obtaining offsets.

Computing Boolean operations among polygons is a well-established research area. Our own implementation, CAPOP, uses a modified Bentley-Ottmann approach [1] to compute the union, intersection, or difference of two regions of $\mathbb{R}^2$ that are bounded by circular-arc polygons. We refer to Martínez et al. [7] for a detailed description of the main algorithm. Although their algorithm can perform Boolean operations only on purely polygonal data, all of the underlying events can be handled analogously in the case of circular arcs. The only important aspect is that all circular arcs that show up in the input need to be broken up into pieces that are monotone in a preprocessing step: If the sweep employed by the Bentley-Ottmann algorithm moves from bottom to top, then every circular arc and every full circle needs to be split at its south pole and at its north pole by inserting up to two new vertices.

For some applications it may be required to distinguish between the inner and the outer offset curves of a polygon P. Of course, the inner offset curves could be identified by subtracting $\mathcal{OA}_v(P, r)$ from the area bounded by P. However, this would require a second run of the Boolean-operation code. Classifying closed loops of the offsets by running point-in-polygon tests for each loop could be equally costly.

Fortunately there is a considerably simpler solution to this problem: For the two straight-line segments of the offset of a variably-weighted straight-line segment $\overline{pq}$ we maintain a pointer to $\overline{pq}$. Similarly we maintain a pointer to the appropriate vertex, p or q, for the two circular caps of that offset. Of course, if an offset segment is split into two segments during the Bentley-Ottmann plane sweep, then these two new segments inherit the pointers from the original segment. Same for arcs.

Once CAPOP has completed the union computations, we loop over all resulting closed offset curves and determine for every offset curve whether it is inside or outside of the input area defined by P. For every offset curve C this decision is made by determining the position of one segment or arc (selected arbitrarily from C) relative to the input segment referenced by its pointer. (For an offset arc we inspect the two input segments that share the input vertex referenced by the pointer.)

Experimental Evaluation:

We evaluated the run-time performance of CAPOP and compared it to several other software packages that support the computation of constant-radius or even variable-radius offsets:

- The Computational Geometry Algorithms Library (CGAL) [8] supports Boolean operations on circular-arc polygons through its `Arrangement_2` package. Thus, by using CGAL it is possible to derive constant-radius as well as variable-radius offsets from polygons.

- The Boost C++ Library (Boost) [6] makes it possible to generate variable-radius offsets by using its geometry buffer functions. However, Boost approximates circular arcs in the input by polygonal chains. By default, Boost approximates a full circle by a polygonal chain consisting of ninety straight-line segments. (Circular arcs are approximated accordingly.)
- Clipper2 [5] supports Boolean operations on simple polygons. Additionally, Clipper2 offers a package for deriving constant-radius offsets of polygons. Similar to Boost, Clipper2 needs to approximate input circular arcs by polygonal chains; we used the same approximation as for Boost.
- Vroni [3] is able to compute Voronoi diagrams of points, straight-line segments, and circular arcs. Based on a Voronoi diagram it can also compute a family of constant-radius offset curves.
- OpenVoronoi [9] is an open-source library that is able to compute Voronoi diagrams of points and straight-line segments. Similar to Vroni, it allows to compute constant-radius offset curves.

In order to be able to include all packages in our test runs we started with timing the computation of constant-radius offsets. The input polygons were taken from the Salzburg Database of Polygonal Data [2]. We tested all packages on about 2700 different polygonal areas with and without holes. All tests were carried out on an Intel Core i9-10980XE processor clocked at 3.0 GHz. In order to avoid an excessively long duration of our tests we set a time limit of 15 minutes for the processing of one input polygon.

The actual offset distance δ is chosen according to an empirically derived formula that takes into account the size of the bounding box of the polygon and the number of its vertices. Thus, the specific value of δ depends on the actual geometry of the input polygon and varies among the different inputs.

Figure 3 shows the run-time results for offset distance δ. In the plot the x-axis corresponds to the number n of vertices of the input polygons, and the y-axis corresponds to the run-time divided by the factor $n \log n$. Both axes are in logarithmic scale. Note that for our CAPOP (apx) test runs we approximated every input circle by a polygonal chain consisting of ninety straight-line segments. Thus, the inputs for CAPOP (apx) were identical to those used by Clipper2 and Boost. Since they approximate input arcs it is needless to say that Clipper2, Boost and CAPOP (apx) cannot compute the correct offset but only a polygonal approximation of the correct offset.

The plot indicates that Vroni, CAPOP and OpenVoronoi are the only packages whose run-times seem to be of the order $O(n \log n)$ for this test setting. Studying a similar plot where all run-times are divided by n^2 suggests that CGAL, Clipper2 and Boost have a (close-to) quadratic complexity. We note, though, that their run-times started to exceed the 15-minute time limit set for one process once n got larger than about 10 000. And none of them managed to handle input files with substantially more than 100 000 vertices within that time limit. OpenVoronoi ended up looping for about eleven percent of our test inputs. CAPOP (apx) ran out of memory for inputs with more than roughly 100 000 vertices. Finally, we note that CGAL is the only code in our tests that used exact arithmetic. Hence, it could be expected to be somewhat slower than the other codes.

In other test series we repeated our runs for offset distance δ scaled by a constant factor, and we timed variable-radius offsets. These additional tests did not produce strikingly different results, and we omit details due to lack of space.

Acknowledgements:
This work was supported by the Austrian Science Fund (FWF): Grant P31013.

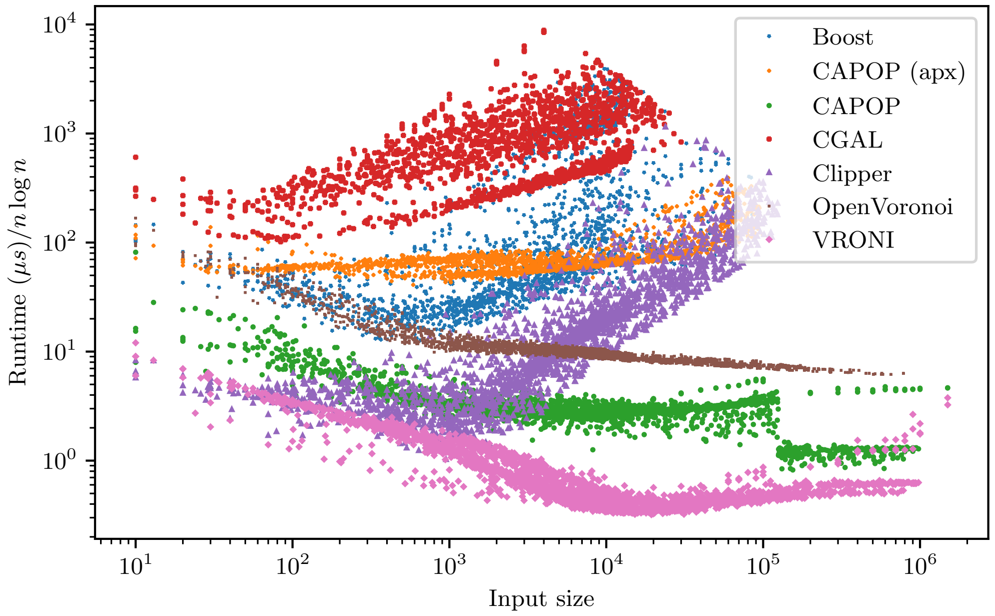


Fig. 3: Experimental run-time results for offset distance δ.

References:

[1] Bentley, J.L.; Ottmann, T.A.: Algorithms for Reporting and Counting Geometric Intersections. IEEE Trans. Comput., C-28(9), 643–647, 1979. http://doi.org/10.1109/TC.1979.1675432.

[2] Eder, G.; Held, M.; Jasonarson, S.; Mayer, P.; Palfrader, P.: Salzburg Database of Polygonal Data: Polygons and Their Generators. Data in Brief, 31, 105984, 2020. ISSN 2352-3409. http://doi.org/10.1016/j.dib.2020.105984.

[3] Held, M.: VRONI and ArcVRONI: Software for and Applications of Voronoi Diagrams in Science and Engineering. In Proc. 8th Int. Symp. Voronoi Diagrams in Science & Engineering, 3–12. IEEE, 2011. http://doi.org/10.1109/ISVD.2011.9.

[4] Held, M.; Huber, S.; Palfrader, P.: Generalized Offsetting of Planar Structures using Skeletons. Comput. Aided Design & Appl., 13(5), 712–721, 2016. http://doi.org/10.1080/16864360.2016.1150718.

[5] Johnson, A.: Clipper2. http://www.angusj.com/clipper2/Docs/Overview.htm.

[6] Koranne, S.: Boost C++ Libraries. In Handbook of Open Source Tools, 127–143. Springer-Verlag, 2011.

[7] Martínez, F.; Rueda, A.J.; Feito, F.R.: A New Algorithm for Computing Boolean Operations on Polygons. Comput. Geosci., 35(6), 1177–1185, 2009. http://doi.org/10.1016/j.cageo.2008.08.009.

[8] The CGAL Project: CGAL User and Reference Manual. CGAL Editorial Board, 5.0 ed., 2019. https://doc.cgal.org/5.0/Manual/packages.html.

[9] Wallin, A.: OpenVoronoi. https://github.com/aewallin/openvoronoi.