Title:
Exact Signed Distance Function Representation of Polygons

Authors:
Csaba Bálint, csabix@inf.elte.hu, Eötvös Loránd University, Budapest
Gábor Valasek, valasek@inf.elte.hu, Eötvös Loránd University, Budapest
Róbert Bán, rob.ban@inf.elte.hu, Eötvös Loránd University, Budapest

Keywords:
Signed Distance Function, Implicit Representation, Geometric Modeling, Collision Detection

DOI: 10.14733/cadconfP.2022.292-296

Introduction:

Signed distance functions (SDF) are applied from high-quality text rendering [3] to geometric representation for collision detection [5], 3D printing, additive manufacturing [1], or advanced real-time graphics effects [7]. The SDF is usually stored as a regular grid of samples for high-performance applications, but various spatial subdivision or interpolation schemes have been proposed for storage, such as octrees [2] or hierarchical T-meshes [6]. In complex shapes, applications mainly focus on storing a discrete approximation to the exact SDF in conjunction with various interpolation techniques.

We propose a conservative but exact SDF representation for planar polygons. The exact SDF is composed of two classes of regions, separated by parabolic and linear boundaries. We construct conservative polygonal bounds to these regions. Our algorithm performs a series of cuts to determine the bounding polygons that represent the distance function on the region. The exact SDF can be evaluated using these polygons. Such a formulation is closely related to point and segment Voronoi diagrams [4]; however, our goal is to preserve the inside-outside partitioning of the plane as well.

Voronoi interpretation:

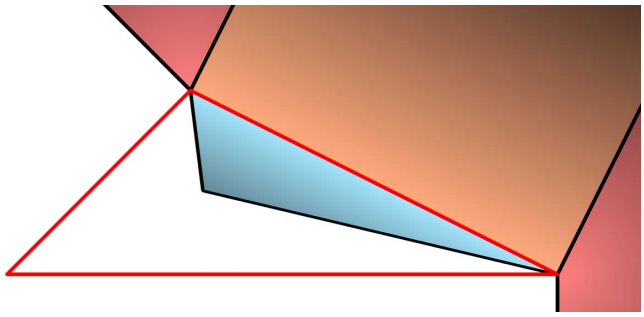
Let us consider an arbitrary polygon, defined by its vertices $\boldsymbol{v}_i$, $i = 1, \ldots, N$. Let $e_i = \{(1-t)\boldsymbol{v}_i + t\boldsymbol{v}_{i+1} \mid t \in (0,1)\}$ denote the open line segments between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$, where $\boldsymbol{v}_{N+1} = \boldsymbol{v}_1$. Then the SDF of a polygon is interpreted as the extension of Voronoi regions to line segments, that is, into two types of sets:

$$V_i = \left\{\boldsymbol{x} \in \mathbb{E}^2 \;\middle|\; \forall j \in \{1, \ldots, N\} \;:\; \|\boldsymbol{x} - \boldsymbol{v}_i\|_2 \leq d(\boldsymbol{x}, e_j)\right\} \subseteq \mathbb{E}^2, \tag{1}$$

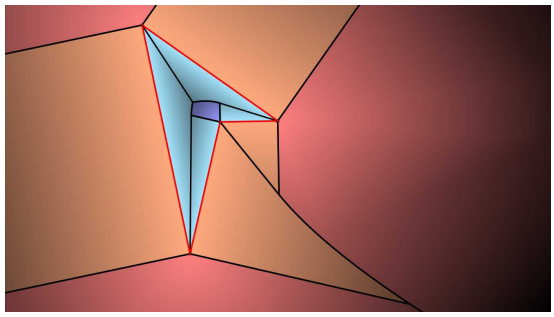
$$E_i = \left\{\boldsymbol{x} \in \mathbb{E}^2 \;\middle|\; \forall j \in \{1, \ldots, N\} \;:\; d(\boldsymbol{x}, e_i) \leq d(\boldsymbol{x}, e_j)\right\} \setminus (V_i \cup V_{i+1}) \subseteq \mathbb{E}^2, \tag{2}$$

where $d(\boldsymbol{x}, A) = \inf_{\boldsymbol{a} \in A} \|\boldsymbol{x} - \boldsymbol{a}\|_2$ denotes the point-to-set distance.

We refer to V_i as a vertex region and E_i as an edge region. Fig. 1a. depicts V_1, E_1, and V_2 regions in a triangle. Every point in an edge region is strictly closer to its closest edge than to any vertex. Note that there are $2N$ regions: N vertex and N edge regions for a polygon with N vertices. Although vertex regions can either lie inside or outside the polygon, edge regions must span across the edge segment. The regions may be computed naively by cutting and combining all of these regions.



(a) The SDF of an edge effects three regions, two corner and a linear edge region with increasing SDF values from the inside negative values to positive outside values.



(b) The exact signed distance function and Voronoi diagram of a concave quadrilateral. Note the parabolic boundaries on the right and inside.

Fig. 1: The exact signed distance function (SDF) of a line segment within a triangle (a) and a concave quadrilateral (b). Vertex regions are red and blue and line regions are orange and sky-blue colored. Distance-based coloring highlights their linear and circular nature.

This paper constructs convex bounding polygons for V_i and E_i for fast signed distance value computation. This yields overlaps, i.e., the regions do not cover the plane uniquely, but the true SDF at any point is obtained by evaluating the finite number of overlapping regions and using the minimum value.

Alternatively, one could intersect the overlapping polygons to obtain an overlap-free polygon covering approximating the parabolic intersections. Another option is storing the exact equations of the perimeter of the corresponding sides to define the region implicitly.

Signed distance function of polygons:

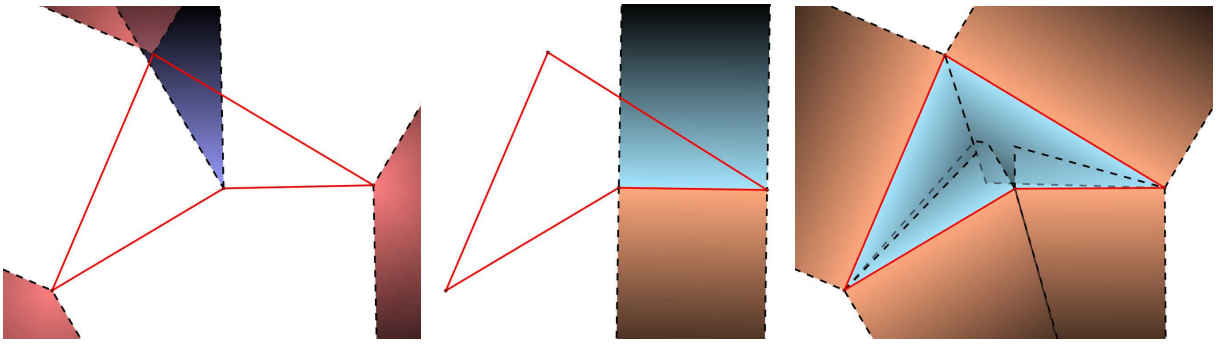
Let us consider a line segment between points $\boldsymbol{v}_i = [x_i, y_i]^T, \boldsymbol{v}_{i+1} = [x_{i+1}, y_{i+1}]^T \in \mathbb{E}^2$ and its SDF. The plane is partitioned into three regions, depending on whether $\boldsymbol{v}_i$, $\boldsymbol{v}_{i+1}$, or an interior point of the line segment is closest. Then, the exact distance function $f : \mathbb{E}^2 \to \mathbb{R}_0^+$ is

$$f(\boldsymbol{x}) = \begin{cases} \|\boldsymbol{x} - \boldsymbol{v}_i\|_2, & \text{if } \boldsymbol{x} \in V_i \\ \|\boldsymbol{x} - \boldsymbol{v}_{i+1}\|_2, & \text{if } \boldsymbol{x} \in V_{i+1} \\ |\boldsymbol{x}^T \boldsymbol{n}_i|, & \text{if } \boldsymbol{x} \in E_i \end{cases}, \qquad \boldsymbol{n}_i = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{\boldsymbol{v}_{i+1} - \boldsymbol{v}_i}{\|\boldsymbol{v}_{i+1} - \boldsymbol{v}_i\|_2}, \tag{3}$$

where $\boldsymbol{n}_i \in \mathbb{R}^2$ is the unit normal vector of the segment. The V_i regions from Eq. 1 store the coordinates of $\boldsymbol{v}_i$ and the sign of the SDF within V_i. The edge regions, E_i from Eq. 2, store the linear distance as an implicit line equation. The sign of the linear equation splits all E_i into two partitions: inside and outside. By convention, we consider the negative partition to be the inside. Note that while the SDF within each region is trivial, the region boundaries themselves may be complex or even concave, a combination of piecewise linear and parabolic curves, see Fig. 1b.

For fast SDF rasterization, one can render all polygons using $|f(\boldsymbol{x})|$ as its Z-depth, and the GPU Z-buffer algorithm will correctly decide the polygon visibility, thereby rasterizing using the correct and exact signed distance function values, see Fig. 4.

Once the regions are known, the polygon SDF is stored in two scalars for each region. A vertex SDF possesses two scalar degrees of freedom, the coordinates of the vertex from which the distance is computed. Similarly, an edge SDF only has to encode a normal direction and a constant term.



(a) Vertex regions are initially cut along neighboring edge normals from the vertex. The red regions have positive SDF values, these outside regions have convex angles.

(b) A single edge region contains both inside and outside regions. They are initialized as the region between these perpendicular lines.

(c) Overlapping edge regions after the edge regions have been cut with every other entity. The regions remain convex and the overlap is resolved by the Z-buffer algorithm.

Fig. 2: Although vertex (a) and edge (b) regions are initially unbounded, the sign of the SDF is determined already. The regions remain convex throughout the cutting process.

Convex region construction:

Our algorithm considers each possible pair of polygon entities, that is, combinations of vertices and edges, and gradually computes bounds to their Voronoi regions.

First, our method constructs initial polygonal regions for all vertices and edges of the polygon. The vertex (Fig. 2a.) and edge (Fig. 2b.) regions are bounded by two half-planes that are perpendicular to the adjacent edges and the edge, respectively. We bound the infinite regions with a large axis-aligned box. Within this box, exact SDF queries may be performed.

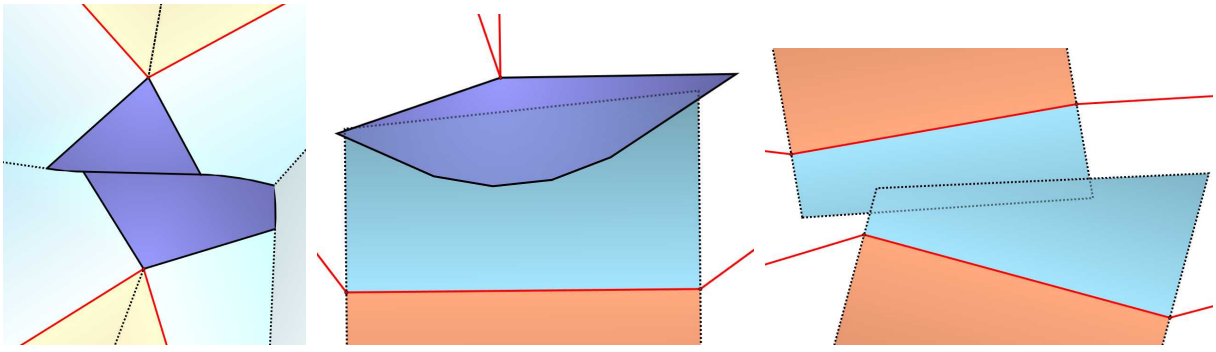
Second, we perform a series of planar cuts to reduce the region sizes to approximate the true one as close as possible. Fig. 2c. depicts the overlapping edge regions after the planar cuts. Note that we need not only intersect the regions of the two current entities; we also have to factor in the cut locus between these two and consider how it splits and overlaps in between the two to form our convex polygonal bounds. Fig. 3. illustrates the vertex-vertex, vertex-edge, and edge-edge cases.

Planar cuts are efficient but only allow convex regions. For each type of cut, we approximate the actual shape with a single bounding cut so that all points are covered, and the SDF can be evaluated everywhere within the bounding box. Our algorithm is summarized as follows:

1. Process input by splitting polygons into boundary components and interior holes. Outer boundaries are processed counterclockwise; hole boundaries are enumerated clockwise to generate correct signs.
2. Initialize each region with its vertex or edge data to reconstruct the distance values, and add two perpendicular cuts using neighbors, see Fig. 2.
3. For each pair of region within each polygon and between neighbouring polygons, perform a planar cut, as in Fig. 3. The following section explains each type of intersection.

Region cuts:

Each vertex generates a circular SDF region on the concave side of the polygon. The polygons of vertex regions undergo a series of cuts. First, they are cut with the perpendicular lines of the adjoining edges going through this point, resulting in regions seen in Fig. 2a.



(a) Vertex regions are cut with each other along their perpendicular bisector, leaving no overlap between vertex regions.

(b) Vertex region is cut with 5 lines approximating the parabola between it and the edge. The edge region is cut with a single line containing the whole parabola segment above the vertex.

(c) Edge regions usually have concave parabolic borders that we bound with a single cut. There are quite a few cases depending on the relative position of the segments that we had to consider.

Fig. 3: We cut regions that are close to each other with lines to retain convex polygonal regions. The three cases, vertex – vertex (a), vertex – edge (b), and edge – edge (c) cases are in increasing complexity.

Second, each region is cut with its neighboring regions in all directions. This may be sped up with a spatial acceleration structure, such as a regular grid, quadtree, or KD-tree, but for small input polygons computing every intersection is still viable. Note that, it is not essential to cut with every other polygon because if we omit distant cuts, the region may be slightly larger, barely affecting rasterization time.

For cutting a vertex region with another, we must cut with the perpendicular bisector in Fig. 3a.

When cutting a vertex region and a line region, the boundary will be parabolic. We approximate these parabolic sides within the vertex region with a fixed number of line segments using tangent cuts, as in Fig. 3b. Although this expands the regions and creates overlaps, it simplifies the algorithm as we only consider linear boundaries. The parabolic cut-locus must be contained within the edge region to retain convexity. Hence, we cut the region with the line that goes through the parabola intersections with the orthogonal lines at the edge endpoints.

Lastly, consider two edge regions as in Fig. 3c. Usually, this would separate the regions along the angular bisector of the two edges on the shorter segment. However, when the footpoint of a cut-locus point becomes an edge endpoint, the boundary becomes parabolic, necessitating complicated formulas that produce the cut. Moreover, neighboring and parallel edges are special cases that we handle differently.

Test results:

The figures in this paper were generated with our Matlab implementation of our proposed algorithm. The regions for the text in Fig. 4. was generated in 2.58 seconds while performing 854124 cuts, and the generation for the tiger took 4.64 seconds with 1575312 polygon cuts.

Conclusions:

We presented an algorithm to compute the exact SDF of polygonal shapes. The runtime of the initial Matlab implementation made it viable as a proxy for computing an SDF partitioning for shapes bounded by parametric curves, such as the ones found in TrueType fonts. The generated SDF regions suggest that expanding the current solution by incorporating circular regions, i.e., by the SDF of a circle, we could make this representation more concise for fonts and other shapes and offsets of polygons. Merging regions

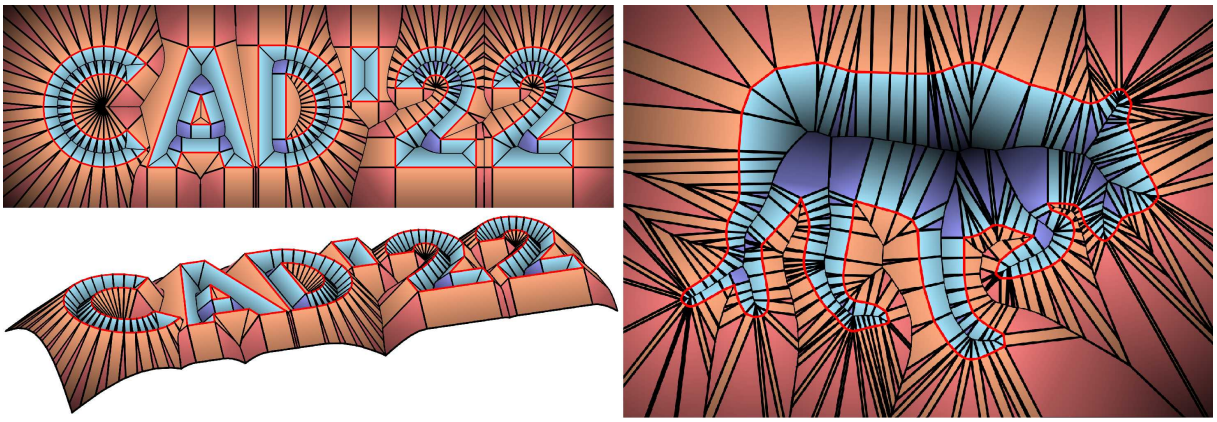


Fig. 4: Generated edge and vertex regions storing exact SDF values of polygons from TrueType fonts from text and a tiger glyph. Overlap between regions are resolved by setting the distance values as third coordinate to indicate depth.

to generate approximate SDF regions within error threshold, expanding it to set-theoretic operations, and optimized generations using bounding volumes or space partitioning is subject to future research.

Acknowledgement:

EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies – The Project is supported by the Hungarian Government and co-financed by the European Social Fund. Supported by the ÚNKP-21-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

References:

[1] Brunton, A.; Rmaileh, L. A.: Displaced Signed Distance Fields for Additive Manufacturing, ACM Trans. Graph. 2021, 40 (4). https://doi.org/10.1145/3450626.3459827

[2] Frisken, S. F.; Perry, R. N.; Rockwood, A. P.; Jones, T. R.: Adaptively Sampled Distance Fields SIGGRAPH '00; ACM Presspp 249–254. https://doi.org/10.1145/344779.344899

[3] Green, C.: Improved Alpha-Tested Magnification for Vector Textures and Special Effects, ACM SIGGRAPH 2007 Courses; SIGGRAPH '07; ACM: San Diego, California, 2007; pp 9–18. https://doi.org/10.1145/1281500.1281665

[4] Held, M.: VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments, Comput. Geom. 18 (2001): 95–123.

[5] Macklin, M.; Erleben, K.; Müller, M.; Chentanez, N.; Jeschke, S.; Corse, Z.: Local Optimization for Robust Signed Distance Field Collision, Proc. ACM Comput. Graph. Interact. Tech. 2020, 3 (1). https://doi.org/10.1145/3384538

[6] Song, X.; Jüttler, B.; Poteaux, A.: Hierarchical Spline Approximation of the Signed Distance Function, 2010 Shape Modeling International Conference; 2010; pp 241–245. https://doi.org/10.1109/SMI.2010.18

[7] Wright, D.: Dynamic Occlusion with Signed Distance Fields, Advances in Real-Time Rendering in Games; SIGGRAPH, 2015.