Journal of Mongolia International University: Research and Innovation. 1(1)2026

Journal of Mongolia International University: Research and Innovation

Journal homepage: https://jri.miu.edu.mn, journal@miu.edu.mn

Research paper

Fast Differentiable Polygon Overlap Using Edge-Edge Kernel

Gadiel Josias Pugh[1] and Jérémie Schertzer[1]

[1] Computer Science Department, Mongolia International University

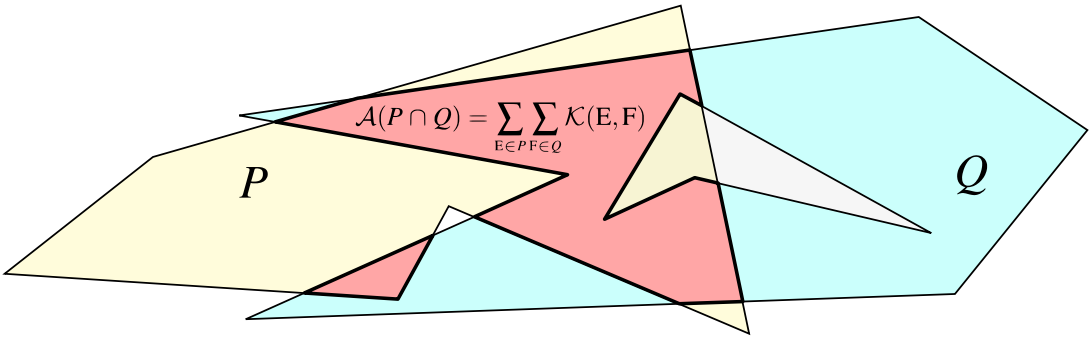
Journal of Mongolia International University: Research and Innovation (2026), Vol. I, No. 1, pp. 34–40

doi: 10.66712/jmri.v1i1.11

Published online: 22 May 2026

Computer Science Department

ABSTRACT



Analytical intersection overlap area (in red) formula between concave polygon P and holed concave polygon Q, using our edge-edge intersection Kernel $\mathcal{K}$.

In this paper, we introduce a differentiable geometry framework for computing polygon overlap areas. Our approach relies on exploiting an existing representation of polygons through edge-based border functions, which collectively define the interior of the shape. Using this formulation, the area of a polygon is expressed as a sum over each edge. For pairs of polygons, the intersection overlap area is derived from simple edge-to-edge kernel functions, defined by integrating border contributions. In addition, our technique can be used to compute differentials of the overlap with respect to polygon vertex coordinates. Our formulation results in a simple branchless expression that naturally supports concave and holed polygons.

Keywords: Computer graphics, Computational geometry, Polygon intersection, Differentiable polygon overlap

1. Introduction

Computing the area of overlap between two polygons is a common operation in computational geometry, with applications for anti-aliasing, vector graphics, collision management... While the area of an individual polygon can be evaluated exactly using the shoelace formula, overlap computation is commonly addressed through polygon clipping algorithms that explicitly construct the intersection shape. These methods are robust and general, but rely on discrete topological decisions and conditional logic, leading to algorithmic complexity that is superfluous when only the overlap value is required. By nature, such approaches do not support analytic (auto-)differentiation with respect to vertex coordinates, which limits the exploitation of overlap in learning frameworks based on loss functions.

In this paper, we introduce an analytical formulation for polygon overlap based on edge border functions, expressing the intersection area as a sum of closed-form edge-edge contributions. This formulation avoids explicit clipping while supporting concave and holed polygons without special treatment, and yields differentiable expressions for overlap with respect to polygon vertices.

In Section 2, we review existing polygon area and clipping approaches, introducing the existing edge bor-

Table 1: Notations

General	
P	Polygon, represented as a list of edges
$\mathcal{A}(P)$	Area of polygon P
$\boxed{P}$	AABB Bounding box of polygon
C_P	edges Count of polygon P
$\mathbf{v}$ $\mathbf{v}_x, \mathbf{v}_y$	2D vertex and its components
$\mathrm{E}(\mathbf{a} \rightarrow \mathbf{b})$	Oriented edge (start→end vertices)
$\mathbf{n}_\mathrm{E}$	Oriented normal of edge E
Operators	
$P \cap Q$	Intersection
$P \cup Q$	Union
$P \setminus Q$	Subtraction
$P \oplus Q$	Exclusive union
Functions	
$\mathbb{1}_P(x, y)$	Indicator function for polygon P
$\mathbb{B}_\mathrm{E}(x, y)$	Border function for E (Section 3.1)
$\mathcal{S}_y(\mathrm{E})$	y-span interval for edge E
$\mathcal{K}(\mathrm{E}, \mathrm{F})$	Edge-edge intersection kernel (Section 3.3)

der function we exploit in Section 3 to formulate our analytical intersection overlap. In Section 4, we discuss our technique and some applications before concluding.

We describe the notation we use in the paper in Table 1.

2. Previous works

The common approach to compute the area of a polygon is set by the shoelace formula [Lim17] (also called the Gauss formula), which provides an exact area calculation based on vertex coordinates. For a polygon P, we have:

$$\mathcal{A}(P) = \frac{1}{2}\left|\sum_{\mathrm{E}(\mathbf{a}\rightarrow\mathbf{b})\in P} \mathbf{a}_x\mathbf{b}_y - \mathbf{a}_y\mathbf{b}_x\right| \tag{1}$$

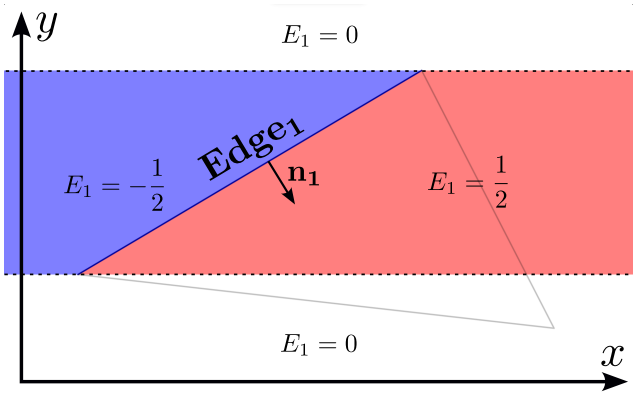
To compute the intersection area between two polygons, current techniques firstly find and define the intersection polygon, and apply the shoelace formula on that polygon.

The Sutherland–Hodgman algorithm [SH74] requires one of the polygons to be convex in order to compute the intersection polygon. More versatile algorithms that can handle general concave shapes are the Weiler–Atherton algorithm [Set03] and the Vatti clipping algorithm [Vat92]. These methods work by identifying intersection points using scanlines, determining in–out regions, and then tracing the new boundary vertices to build the intersection polygon. These algorithms are foundational in polygon processing libraries and software.

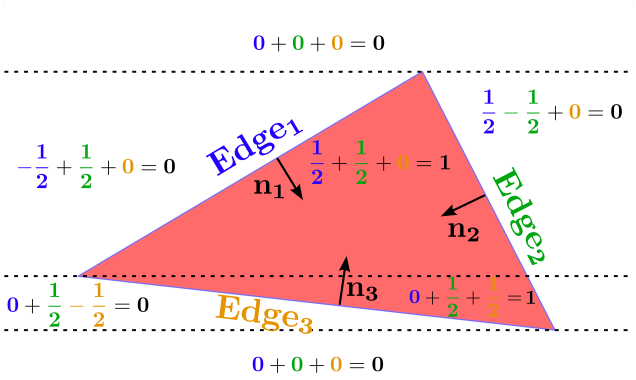
In the context of this paper, these methods suffer from performance limitations and complexity, as they solve the intersection polygon, which we show to be unnecessary when computing only the overlap. Since the clipping relies on discrete events and topology, rather than an analytical closed form, these techniques do not offer the capability of differentiation with respect to vertex coordinates.

Closer to our contribution, Schertzer *et al.* [STB23] provided a closed-form solution for the overlap between a polygon and a square. From a signal processing perspective, they introduced edge border functions ($\mathbb{B}_\mathrm{E}(x, y)$, Figure 2a) to define an indicator function ($\mathbb{1}_P(x, y)$, Figure 2b) for a polygon P:

$$\mathbb{1}_P(x, y) = \sum_{\mathrm{E}\in P} \mathbb{B}_\mathrm{E}(x, y) \tag{2}$$



(a) Visualization of border function $\mathbb{B}_\mathrm{E}(x, y)$



(b) Visualization of polygon indicator function $\mathbb{1}_P(x, y)$ as the sum of edge border functions.

Figure 2: Illustration of functions introduced by Schertzer *et al.*. Figure taken and used with authorization from [STB23].

From that indicator function, the overlap between P and an axis-aligned square Q of size a, centered at x_0, y_0, is computed using convolutions, leveraging the fact that a square behaves as a separable gate kernel:

$$\mathcal{A}(P \cap Q) = [\mathbb{1}_P * gate_x * gate_y](x_0, y_0) \quad (3)$$
$$= \sum_{\mathrm{E} \in P} [\mathbb{B}_\mathrm{E} * gate_x * gate_y](x_0, y_0) \quad (4)$$

By providing a closed form of the convoluted edge border function, the overlap is computed as a simple sum, as Equation (4) shows. Note that our contribution relies on and exploits their formulation defined in Equation (2). We present edge border functions precisely in Section 3.1.

Finally, polygon representation using signed distance fields [BVB23] excels at rendering intersection overlap, but fails to provide a value of that overlap.

3. Polygon intersection area

In this section, we describe our polygon overlap closed-form formula, starting from the edge border function formulation for polygons.

3.1. Definitions and hypotheses

We consider that polygons may include holes and concavities, but without self-intersection.

For an oriented edge $\mathrm{E}(\mathbf{a} \to \mathbf{b})$, we define the y-span as the interval where the edge lies, in the y dimension:

$$\mathcal{S}_y(\mathrm{E}) = [\min(\mathbf{a}_y, \mathbf{b}_y); \max(\mathbf{a}_y, \mathbf{b}_y)] \quad (5)$$

Let $\mathbf{n}_\mathrm{E}$ be an oriented normal vector to the edge $\mathrm{E}(\mathbf{a} \to \mathbf{b})$. We assume that polygon normals all follow the same orientation convention, either pointing inward or outward. Note that our method does not require normalizing normals, which we compute directly as follows:

$$\mathbf{n}_\mathrm{E} = \begin{pmatrix} (\mathbf{b}-\mathbf{a})_y \\ -(\mathbf{b}-\mathbf{a})_x \end{pmatrix} = \begin{pmatrix} \mathbf{b}_y - \mathbf{a}_y \\ \mathbf{a}_x - \mathbf{b}_x \end{pmatrix} \quad (6)$$

For the rest of the paper, we consider $\mathbf{v} \in \mathbb{R}^2$, a vector of coordinates x and y. When using functions with input in $\mathbb{R}^2$, we either use $\mathbf{v}$ or its coordinates (x, y), depending on what is more convenient.

For an oriented edge $\mathrm{E}(\mathbf{a} \to \mathbf{b})$, Schertzer *et al.* defined the border function $\mathbb{B}_\mathrm{E}$ as follows:

$$\mathbf{v} \in \mathbb{R}^2 \to \mathbb{B}_\mathrm{E}(\mathbf{v}) \in \{-\tfrac{1}{2}, \tfrac{1}{2}\}$$
$$\mathbb{B}_\mathrm{E}(\mathbf{v}) = \begin{cases} \frac{1}{2}\, sign((\mathbf{v}-\mathbf{a}) \cdot \mathbf{n}_\mathrm{E}) & , y \in \mathcal{S}_y(\mathrm{E}) \\ 0 & , \text{else} \end{cases} \quad (7)$$

Using the indicator function of a polygon P, the formulation of its area directly appears as an integral over $\mathbb{R}^2$:

$$\mathcal{A}(P) = \iint_{\mathbb{R}^2} \mathbb{1}_P(x, y)\, dx\, dy \quad (8)$$

Since the domain where $\mathbb{1}_P \neq 0$ is finite, the infinite integral can be reduced to $\boxed{P}$, the bounding box of P:

$$\mathcal{A}(P) = \iint_{\boxed{P}} \mathbb{1}_P(x, y)\, dx\, dy \quad (9)$$

3.2. Formulation of overlap

In this section, we define the intersection area as the resulting area from the intersection polygon between two polygons P and Q. We demonstrate that this overlap can be expressed by summing edge-edge kernel functions $\mathcal{K}$:

$$\mathcal{A}(P \cap Q) = \sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \mathcal{K}(\mathrm{E}, \mathrm{F}) \quad (10)$$

The overlap is defined using the product of indicator functions, which can be decomposed with border functions using Equation (2):

$$\mathbb{1}_{P \cap Q}(\mathbf{v}) = \mathbb{1}_P(\mathbf{v}) \cdot \mathbb{1}_Q(\mathbf{v})$$
$$\mathbb{1}_{P \cap Q}(\mathbf{v}) = \sum_{\mathrm{E} \in P} \mathbb{B}_\mathrm{E}(\mathbf{v}) \cdot \sum_{\mathrm{F} \in Q} \mathbb{B}_\mathrm{F}(\mathbf{v})$$
$$\mathbb{1}_{P \cap Q}(\mathbf{v}) = \sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \mathbb{B}_\mathrm{E}(\mathbf{v}) \mathbb{B}_\mathrm{F}(\mathbf{v}) \quad (11)$$

By combining Equation (9) with Equation (11), we have:

$$\mathcal{A}(P \cap Q) = \iint_{\boxed{P \cup Q}} \mathbb{1}_{P \cap Q}(x, y)\, dx\, dy \ , \text{ since } \boxed{P \cap Q} \subset \boxed{P \cup Q}$$
$$\mathcal{A}(P \cap Q) = \iint_{\boxed{P \cup Q}} \left(\sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \mathbb{B}_\mathrm{E}(x, y) \mathbb{B}_\mathrm{F}(x, y) \right) dx\, dy$$
$$\mathcal{A}(P \cap Q) = \sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \iint_{\boxed{P \cup Q}} \mathbb{B}_\mathrm{E}(x, y) \mathbb{B}_\mathrm{F}(x, y)\, dx\, dy \quad (12)$$

From Equation (12), we finally identify $\mathcal{K}$:

$$\mathcal{K}(\mathrm{E}, \mathrm{F}) = \iint_{\boxed{P \cup Q}} \mathbb{B}_\mathrm{E}(x, y) \mathbb{B}_\mathrm{F}(x, y)\, dx\, dy \quad (13)$$

3.3. Expression of intersection kernel

In this section, we demonstrate that the intersection kernel $\mathcal{K}$ can be reduced to a closed form.

As a first step, we rearrange the integration domain defining $\mathcal{K}$. We call L the x-span of $\boxed{P \cup Q}$.

We also exploit the fact that the non-zero domain of $\mathbb{B}_\mathrm{E}(x, y)\, \mathbb{B}_\mathrm{F}(x, y)$ is reduced to $y \in \mathcal{S}_y(\mathrm{E}) \cap \mathcal{S}_y(\mathrm{F}) = [m, M]$. Note that this domain is null when the two edges have no common y-span. In that simple case, $\mathcal{K}(\mathrm{E}, \mathrm{F}) = 0$.

From $\mathrm{E}(\mathbf{a} \to \mathbf{b})$ and $\mathrm{F}(\mathbf{c} \to \mathbf{d})$, m and M are com-

puted as follows:

$$m = \max(\min(\mathbf{a}_y, \mathbf{b}_y), \min(\mathbf{c}_y, \mathbf{d}_y)) \quad (14)$$

$$M = \min(\max(\mathbf{a}_y, \mathbf{b}_y), \max(\mathbf{c}_y, \mathbf{d}_y)) \quad (15)$$

If $m \geq M$, then $\mathcal{K}(\mathrm{E}, \mathrm{F}) = 0$.

From these considerations, Equation (13) can be written as:

$$\mathcal{K}(\mathrm{E}, \mathrm{F}) = \int_m^M \int_L \mathbb{B}_\mathrm{E}(x, y)\mathbb{B}_\mathrm{F}(x, y)\, dx\, dy \quad (16)$$

Over that domain, $\mathbb{B}_\mathrm{E}(x, y) \neq 0$ and $\mathbb{B}_\mathrm{F}(x, y) \neq 0$, so the expression of border functions can be reinserted:

$$\mathcal{K}(\mathrm{E}, \mathrm{F}) = \frac{1}{4} \int_m^M \int_L sign((\mathbf{v} - \mathbf{a}) \cdot \mathbf{n}_\mathrm{E})\, sign((\mathbf{v} - \mathbf{c}) \cdot \mathbf{n}_\mathrm{F})\, dx\, dy \quad (17)$$

The integrand of Equation (17) features interesting properties, as it shapes a gate function of x, stepping at $X_0(y)$ and $X_1(y)$.

Assuming non-degenerate cases, we solve:

$$X_0(y) = \frac{\mathbf{a} \cdot \mathbf{n}_\mathrm{E} - y \cdot \mathbf{n}_{\mathrm{E}y}}{\mathbf{n}_{\mathrm{E}x}} \quad (18)$$

$$X_1(y) = \frac{\mathbf{c} \cdot \mathbf{n}_\mathrm{F} - y \cdot \mathbf{n}_{\mathrm{F}y}}{\mathbf{n}_{\mathrm{F}x}} \quad (19)$$

We compute the inner dx integral using basic rectangle geometry, since we have a gate function:

$$\begin{aligned} &\int_L sign((\mathbf{v} - \mathbf{a}) \cdot \mathbf{n}_\mathrm{E})\, sign((\mathbf{v} - \mathbf{c}) \cdot \mathbf{n}_\mathrm{F})\, dx \\ &= sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x}) \cdot (L - 2\,|X_1(y) - X_0(y)|) \\ &= sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x}) \cdot (L - 2\,|\alpha \cdot y - \beta|) \\ &\text{with } \alpha = \frac{\mathbf{n}_{\mathrm{E}y}}{\mathbf{n}_{\mathrm{E}x}} - \frac{\mathbf{n}_{\mathrm{F}y}}{\mathbf{n}_{\mathrm{F}x}},\ \beta = \frac{\mathbf{a} \cdot \mathbf{n}_\mathrm{E}}{\mathbf{n}_{\mathrm{E}x}} - \frac{\mathbf{c} \cdot \mathbf{n}_\mathrm{F}}{\mathbf{n}_{\mathrm{F}x}} \end{aligned} \quad (20)$$

From this expression, $\mathcal{K}(\mathrm{E}, \mathrm{F})$ is found using the antiderivative of the absolute value function:

$$\mathcal{K} = \frac{sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x})}{4}(M - m) \cdot L + \frac{sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x})}{4} \frac{(\alpha \cdot m - \beta)|\alpha \cdot m - \beta| - (\alpha \cdot M - \beta)\,|\alpha \cdot M - \beta|}{\alpha} \quad (21)$$

Equation (21) displays a closed form for the intersection kernel in the non-degenerate cases. In practice, the degenerate cases have even simpler expressions described in Section 3.4.

Finally, we consider a further simplification for $\mathcal{K}(\mathrm{E}, \mathrm{F})$, relying on geometric properties canceling out. Using the integral over closed contours, it appears that:

$$\sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \frac{sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x})}{4}(M - m) \cdot L = 0 \quad (22)$$

This was expected, as L was chosen arbitrarily, since we could integrate on a finite domain larger than $\boxed{P \cup Q}$.

This leads to the final expression of $\mathcal{K}(\mathrm{E}, \mathrm{F})$, dropping a term:

$$\frac{sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x})}{4} \frac{(\alpha \cdot m - \beta)|\alpha \cdot m - \beta| - (\alpha \cdot M - \beta)\,|\alpha \cdot M - \beta|}{\alpha} \quad (23)$$

Our overlap formulation results in a robust expression using purely algebraic operations on vertex coordinates of the polygons.

3.4. Degenerate cases

Our intersection kernel formula from Equation (23) implies that $\mathbf{n}_{\mathrm{E}x} \neq 0$ and $\mathbf{n}_{\mathrm{F}x} \neq 0$. Indeed, if any of those values is equal to zero, the edge is horizontal, resulting in $m = M$, thus $\mathcal{K}(\mathrm{E}, \mathrm{F}) = 0$.

Still, one degenerate case where $\alpha = 0$ can exist, which is resolved as follows:

$$\alpha = 0 \implies \mathcal{K}(\mathrm{E}, \mathrm{F}) = \frac{sign(\mathbf{n}_{\mathrm{E}x} \cdot \mathbf{n}_{\mathrm{F}x})}{2} \cdot |\beta| \cdot (m - M) \quad (24)$$

3.5. Extended overlaps

From the intersection area values, other overlaps (union, xor, subtraction) can be trivially computed by combining the value of the area of each polygon.

Using the basic property of sets, the union overlap can be derived using intersection overlap:

$$\mathcal{A}(P \cup Q) = \mathcal{A}(P) + \mathcal{A}(Q) - \mathcal{A}(P \cap Q) \quad (25)$$

$$\mathcal{A}(P \cup Q) = \mathcal{A}(P) + \mathcal{A}(Q) - \sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \mathcal{K}(\mathrm{E}, \mathrm{F}) \quad (26)$$

In a similar fashion, the xor overlap follows the following expression:

$$\mathcal{A}(P \oplus Q) = \mathcal{A}(P) + \mathcal{A}(Q) - 2\mathcal{A}(P \cap Q) \quad (27)$$

$$\mathcal{A}(P \oplus Q) = \mathcal{A}(P) + \mathcal{A}(Q) - 2\sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \mathcal{K}(\mathrm{E}, \mathrm{F}) \quad (28)$$

Finally, the area of the subtraction between P and Q:

$$\mathcal{A}(P \setminus Q) = \mathcal{A}(P) - \mathcal{A}(P \cap Q) \quad (29)$$

$$\mathcal{A}(P \setminus Q) = \mathcal{A}(P) - \sum_{\mathrm{E} \in P} \sum_{\mathrm{F} \in Q} \mathcal{K}(\mathrm{E}, \mathrm{F}) \quad (30)$$

4. Analysis and Applications

4.1. Implementation

Our technique benefits greatly from its simplicity, leading to a straightforward implementation without relying on advanced data structures or sorting algorithms.

Implementation of our formula requires no extra memory apart from storing polygon edges. In order to save some multiplications and divisions, for any edge $\mathrm{E}(\mathbf{a} \rightarrow \mathbf{b})$, the values $\frac{\mathbf{n}_{\mathrm{E}y}}{\mathbf{n}_{\mathrm{E}x}}$ and $\frac{\mathbf{a} \cdot \mathbf{n}_\mathrm{E}}{\mathbf{n}_{\mathrm{E}x}}$ could be precomputed into a table.

Algorithm 1 Python implementation of edge-edge kernel and polygon overlap using SymPy, allowing auto-differentiation

```
def EdgeEdgeKernel(a, b, c, d):
  # input edge E(a→b) and F(c→d)
  # v[0]: x component, v[1]: y component
  # Computation of oriented normals (un-normalized)
  nE = sp.Matrix([b[1]-a[1], a[0]-b[0]])
  nF = sp.Matrix([d[1]-c[1], c[0]-d[0]])

  # Computation of integration boundaries
  m = sp.Max(sp.Min(a[1],b[1]), sp.Min(c[1],d[1]))
  M = sp.Min(sp.Max(a[1],b[1]), sp.Max(c[1],d[1]))
  # Without autodiff, early return zero when m > M

  # Note that nE[0] or nF[0] = 0 => m=N => K=0
  alpha = nE[1]/nE[0] - nF[1]/nF[0]
  beta = a.dot(nE)/nE[0] - c.dot(nF)/nF[0]

  S = sp.sign(nE[0]*nF[0])
  # Kernel in degenerate case when |alpha|<eps
  K_degen = S/2.0 * sp.Abs(beta) * (m - M)

  # Kernel in general case
  K = S/(4.0*alpha) * (
     (alpha*m - beta)*sp.Abs(alpha*m - beta) -
     (alpha*M - beta)*sp.Abs(alpha*M - beta))

  # Return as a differentiable branching
  return sp.Piecewise(
     (0, m >= M - eps),
     (K_degen, sp.Abs(alpha) < eps),
     (K, True))

def PolygonsOverlap(poly1, poly2):
  # Each polygon as a list of edges,
  # Each edge as a pair of 2D vertices
  overlap = sp.Float(0.0)
  for (a, b) in poly1: # Looping on edges (a→b)
    for (c, d) in poly2: # Looping on edges (c→d)
      overlap += EdgeEdgeKernel(a, b, c, d)
  return overlap
```

Also, our technique offers low variance in performance, as operations are independent of polygon configurations (unlike techniques requiring sorting). The only optimization simplifying computations is related to early return $\mathcal{K}(\mathrm{E},\mathrm{F}) = 0$ on disjoint edges when $M < m$.

The direct summation of kernel evaluations fueling our overlap formula makes our technique very GPU/SIMD-friendly, where each intersection kernel can be concurrently evaluated on different threads, followed by an atomic sum to merge the final result.

When considering overlap schemes relying on differentiation, our formula can be smoothly implemented with libraries supporting auto-differentiation (PyTorch, SymPy, . . .). We present in Algorithm 1 the Python implementation we used for applications of our technique, using SymPy for auto-differentiation.

4.2. Asymptotic performance

From Equation (10), it directly appears that for P with C_P edges, and Q with C_Q edges, our analytic overlap computation requires an asymptotic cost of $O(C_P \cdot C_Q)$. This cost does not depend on the topology of the polygons considered.

In comparison, Sutherland–Hodgman clipping also relies on an $O(C_P \cdot C_Q)$ cost, but requires an intermediate growing list storing intersections. Also, the Sutherland–Hodgman technique assumes at least one of the polygons to be convex.

The more general technique introduced by Weiler–Atherton, supporting concavities and holes, presents a better average cost, but with a worst-case scenario of $O(C_P \cdot C_Q)$. It also implies a second pass traversing a linked list with a cost of $O(C_P + C_Q)$.

Finally, Vatti clipping, supporting self-intersections on top of holes and concavities, presents a common cost of $O((C_P + C_Q) \cdot \log(C_P + C_Q))$, which can reach a worst-case cost of $O(C_P \cdot C_Q \cdot \log(C_P + C_Q))$. This technique requires storing a dynamic list of active vertices.

4.3. Differentiable overlap

For scenarios where the target is related to optimizing vertices of a polygon, constrained by the value of an overlap, differentiation of the overlap with respect to vertex coordinates is required to descend the gradient.

We consider P and Q, two polygons, and aim to compute:

$$\frac{\partial \mathcal{A}(P(\mathbf{b}) \cap Q)}{\partial \mathbf{b}}, \text{ where } \mathbf{b} \text{ is a vertex of } P \tag{31}$$

We can find $\mathrm{E1}(\mathbf{a} \to \mathbf{b})$ and $\mathrm{E2}(\mathbf{b} \to \mathbf{c}) \in (P)$ that are the only two edges depending on $\mathbf{b}$, which leads to:

$$\frac{\partial \mathcal{A}(P(\mathbf{b}) \cap Q)}{\partial \mathbf{b}} = \frac{\partial \sum_{\mathrm{F} \in Q} (\mathcal{K}(\mathrm{E1}(\mathbf{b}), \mathrm{F}) + \mathcal{K}(\mathrm{E2}(\mathbf{b}), \mathrm{F}))}{\partial \mathbf{b}} \tag{32}$$

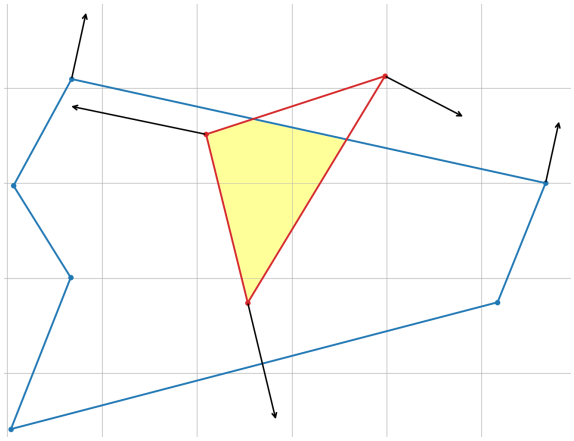
A dual equation can be written for vertices belonging to Q, implying that the cost of computing the gradient $\nabla \mathcal{A}(P \cap Q)$ *w.r.t.* $2 \cdot C_P + 2 \cdot C_Q$ scalar coordinates is $O(C_P \cdot C_Q)$.

We present a visualization of the overlap gradient in Figure 3.

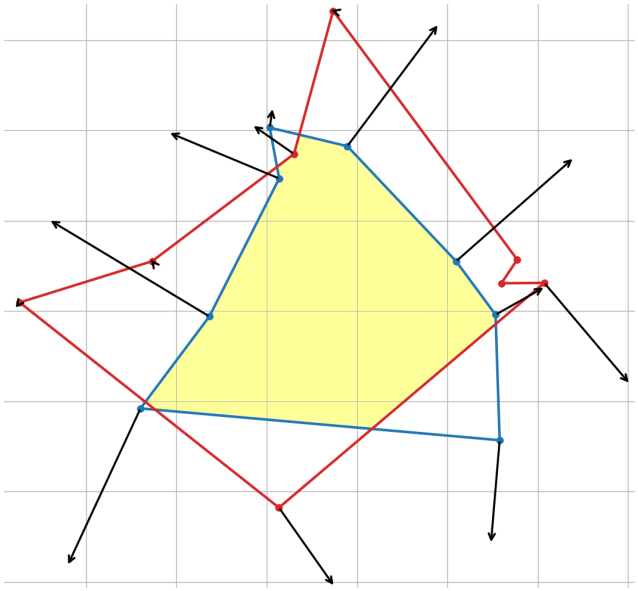
4.4. Anti-aliasing

As covered in the previous work section, computation of the overlap of a polygon with a square has already been resolved, enabling anti-aliasing applications on square or rectangular grids.

Our contribution proposes a generalization to any regular or irregular grid, by computing the overlap between each polygon of the grid and the polygon to be rendered. Figure 4a displays the anti-aliased rendering of a polygon on a hexagonal grid, while Figure 4b features a triangle grid.



(a) Overlap value = 1.4323



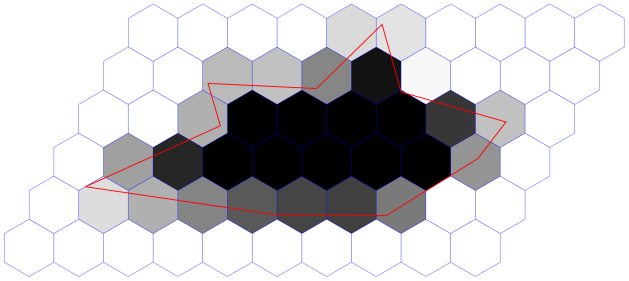
(b) Overlap value = 7.0797

Figure 3: Visualization of overlap gradient *w.r.t.* vertex coordinates of polygons. When non-zero, the gradient is represented with an arrow, indicating the direction to shift vertices to maximize overlap growth. The norm of the gradient gives information on the importance of a vertex shift on the overlap. One square of the grid is equal to one unit of surface.

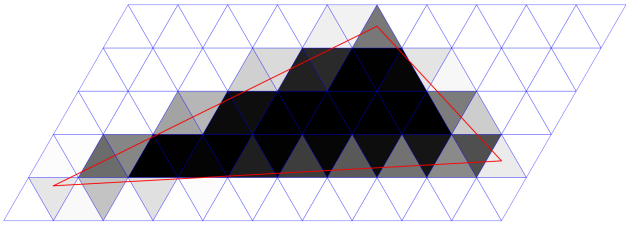
5. Conclusion

5.1. Summary

In this paper, we discussed the limitations of existing techniques aiming to compute polygon intersection overlap. To introduce our contribution, we leveraged the edge-border function introduced by Schertzer *et al.* Using a proper formulation of the overlap with polygon indicator functions, we detailed the expression of an edge-edge kernel function, that we are able to integrate using geometric and analytic considerations. This results in a compact and straightforward expression that can be simply implemented and swiftly evaluated, supporting auto-differentiation.



(a) Anti-aliasing on a hexagonal grid using our overlap formula



(b) Anti-aliasing on a triangle grid using our overlap formula

Figure 4: Intersection overlap for generalized anti-aliasing

5.2. Limitations

By design of our formula, self-intersecting polygons result in computing potentially negative overlaps, as the edge normals are flipped by the self-intersection. Addressing that matter would require splitting polygons, which is achieved using techniques inspired by Vatti clipping, defeating the purpose of computing overlap without an intersection polygon.

Since our technique relies on summing terms canceling each other to compute the overlap, numerical inaccuracy may occur on polygons with large number of edges, as well as on some pathological near-horizontal edges. A conservative solution to avoid summing numbers of different orders of magnitude relies on considering edges degenerate when $|\mathbf{n}_{\mathrm{E}x}| < \epsilon$.

5.3. Future works

By nature, the gradient becomes null when both polygons are disjoint: $\mathcal{A}(P \cap Q) = 0 \implies \nabla\mathcal{A}(P \cap Q) = \mathbf{0}$.

While this is mathematically correct, it hinders gradient-based optimization since it becomes impossible to know where to move vertices to increase overlap. A solution for this issue can be explored by tweaking the part of the edge border function modeling the exterior of an edge. This would lead to a smooth non-zero gradient allowing convergence from disjoint initial conditions. To formalize this idea, the expression of the integrals and the influence of the induced bias have to be properly studied.

For a generalization in three dimensions, we believe that a triangle-triangle function can be defined and used

to build a volume intersection overlap kernel between two triangle meshes. Such a 3D intersection kernel would imply three nested integrals that may raise some challenges.

Finally, our intersection overlap may benefit visibility testing for differentiable rendering as well as collision management between polygons.

References

[BVB23] Bálint C., Valasek G., Bán R.:
Exact signed distance function representation of polygons.
Comput-Aided Des. Appl 20 (2023), 1029–1042.

[Lim17] Lim W.:
Shoelace formula: Connecting the area of a polygon and vector cross product.
Mathematics Teacher 110 (04 2017), 631–636.
doi:10.5951/mathteacher.110.8.0631.

[Set03] Setiabudi D. H.:
Pemotongan poligon menggunakan algoritma weiler atherton.
Jurnal Teknik Elektro Universitas Kristen Petra 3, 1 (2003), 133315.

[SH74] Sutherland I. E., Hodgman G. W.:
Reentrant polygon clipping.
Communications of the ACM 17, 1 (1974), 32–42.

[STB23] Schertzer J., Thonat T., Boubekeur T.:
Triangle rejection sampling for density-equipped meshes on gpu.
Computer Graphics Forum (Proc. High Performance Graphics 2025) 44, 8 (2023).
doi:10.1111/cgf.70217.

[Vat92] Vatti B. R.:
A generic solution to polygon clipping.
Communications of the ACM 35, 7 (1992), 56–63.