Minkowski Penalties: Robust Differentiable Constraint Enforcement for Vector Graphics

JIŘI MINARČÍK, Independent Researcher, Czech Republic
SAM ESTEP, WODE NI, and KEENAN CRANE, Carnegie Mellon University, USA

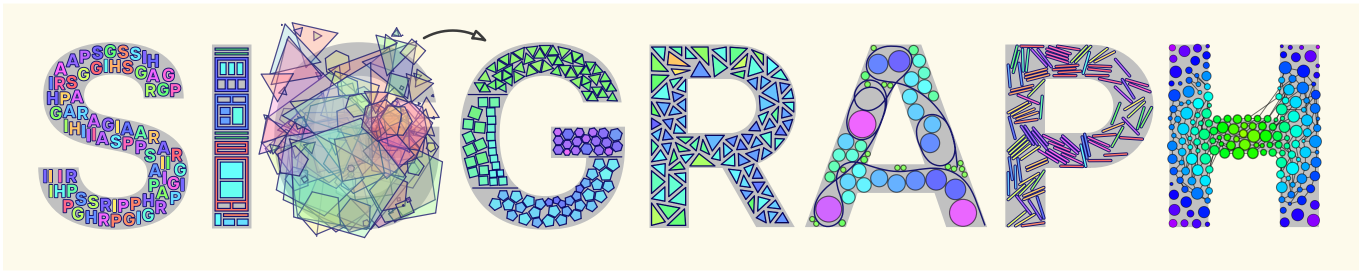


Fig. 1. Minkowski penalties provide robust enforcement of general geometric predicates, for a wide variety of shapes. Here we highlight, from left to right: (S) glyph placement with precise padding, (I) nested containment, (GG) successful optimization from a highly infeasible state, (R) high-quality packing and boundary alignment, (A) joint constraint enforcement and energy optimization, (P) constraint enforcement for shapes that do not have a well-defined inside/outside, and (H) no-overlap constraints integrated with graph layout. In each case (except (I)), shapes are confined to a nonconvex region.

This paper describes an optimization-based framework for finding arrangements of 2D shapes subject to pairwise constraints. Such arrangements naturally arise in tasks such as vector illustration and diagram generation, but enforcing these criteria robustly is surprisingly challenging. We approach this problem through the minimization of novel energetic penalties, derived from the signed distance function of the Minkowski difference between interacting shapes. This formulation provides useful gradients even when initialized from a wildly infeasible state, and, unlike many common collision penalties, can handle open curves that do not have a well-defined inside and outside. Moreover, it supports rich features beyond the basic no-overlap condition, such as tangency, containment, and precise padding, which are especially valuable in the vector illustration context. We develop closed-form expressions and efficient approximations of our penalty for standard vector graphics primitives, yielding efficient evaluation and easy implementation within existing automatic differentiation pipelines. The method has already been "battle-tested" as a component of public-facing open source software; we demonstrate the utility of the framework via examples from illustration, data visualization, diagram generation, and geometry processing.

CCS Concepts: • **Human-centered computing → Visualization**; • **Theory of computation** → *Mathematical optimization.*

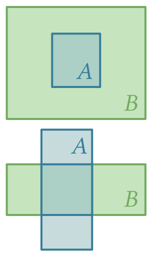
Additional Key Words and Phrases: geometric constraints, mathematical diagrams, Minkowski difference, signed distance function

ACM Reference Format:
Jiři Minarčík, Sam Estep, Wode Ni, and Keenan Crane. 2024. Minkowski Penalties: Robust Differentiable Constraint Enforcement for Vector Graphics. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3641519.3657495

Authors' Contact Information: Jiři Minarčík, Independent Researcher, Czech Republic; Sam Estep; Wode Ni; Keenan Crane, Carnegie Mellon University, Pittsburgh, PA, USA.

1 INTRODUCTION

Layout of 2D shapes is a central task in graphic design, data visualization, diagramming, and document generation. In many situations, it can be framed as a geometric constraint satisfaction problem: find shape parameters that satisfy a given set of layout constraints. While layout strategies are intensely studied for special classes of problems such as graph drawing [Tamassia 2013] or document layout [Hurst et al. 2009], layout of general 2D illustrations is not as well-developed. We approach this problem from the perspective of continuous, descent-based optimization. While other strategies (*e.g.*, random sampling or SAT solvers) are of course quite valuable, a penalty method is easily integrated with existing continuous formulations, and widely-used automatic differentiation/backpropagation frameworks.



As a motivating problem, consider the simple task of arranging two axis-aligned rectangles A and B such that they do not overlap. More precisely, suppose we seek a penalty function $\mathcal{P}$ depending on the parameters of A and B (center, width, and height) such that following the gradient of $\mathcal{P}$ leads to a feasible state. This innocent-sounding problem is deceptively hard to solve. For instance, simply minimizing overlap area is insufficient, since the gradient of this area is zero in many overlapping configurations (inset, top). Likewise, penalizing mutual containment of corners can fail (inset, bottom); more generally, any strategy based on sampling a finite number of points can hence miss overlaps. (Section 5.1 considers this situation in more detail.)

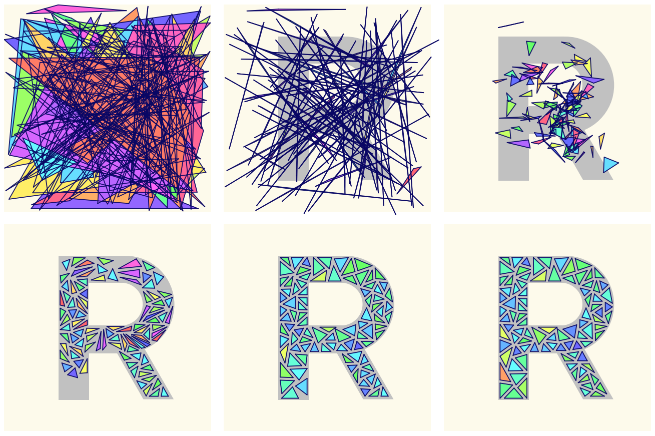


Fig. 2. User-specified illustrations and diagrams can be unpredictable, making it hard to find a feasible initial state. Our method places no such burden on the user, permitting completely random initialization *(top left)*. Here for instance we optimize triangles so that they maximize area and are as equilateral as possible, while requiring them to be mutually disjoint (with a small amount of padding) and contained within the nonconvex letter "R".

Layout of course becomes much harder once we look beyond this toy problem, toward larger classes of shapes and constraints. The challenge is not finding a better optimization scheme, but first and foremost designing penalty functions with the desired behavior. In particular, we seek penalties that meet the following requirements:

- (Min) Global minima correspond to feasible configurations.
- (Diff) Differentiable with respect to shape parameters.
- (Grad) Nonzero gradient if and only if constraints are violated.[1]
- (Shape) Applies to general shapes (nonconvex, open ...).
- (Param) Permits arbitrary, user-defined parameterization of shapes (*e.g.*, polylines encoded as edge lengths and turning angles), rather than just rigid translation.

As explored in Section 5.1, however, it is surprisingly challenging to design penalties that simultaneously satisfy all of these properties.

Our solution builds on the *penetration depth* long used in robotics, adapting this perspective to vector graphics layout. In particular, we express pairwise constraints on shapes A and B in terms of the signed distance function (SDF) ϕ_C of their Minkowski difference $C := A - B$ (Section 3), which is well-defined for arbitrary open and closed curves (Shape). The resulting *Minkowski penalties* are nonnegative, become zero only when constraints are satisfied (Min), and exhibit nonzero gradients whenever constraints are violated (Grad). We couple these penalties with a carefully-chosen *exterior point method* that permits infeasible states. These components are well-matched: the exterior point method avoids the need for feasible initialization, and Minkowski penalties provide useful gradient information even in infeasible states. As a result, we can obtain valid solutions even from a wildly infeasible initialization (Figure 2).

[1]*Isolated* maxima/saddles are acceptable, since numerical perturbations or intensional derivatives [Lee et al. 2020, Section 3] lead descent methods away from such points.

Moreover, since Minkowski penalties are easily integrated with modern autodiff (Diff), they can be used in conjunction with arbitrary shape parameterizations (Param).

1.1 Layout Problem

Consider a collection of shapes $\mathcal{S} = \{S_1, \ldots, S_n\}$, parameterized by degrees of freedom $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{R}^m$ (*e.g.*, centers and radii for circles, or vertices for polylines). We seek a configuration of $\mathcal{S}$ that satisfies a collection of geometric predicates (containment, nonoverlap, *etc.*), encoded by nonnegative *penalty functions* $\mathcal{P}_1, \ldots, \mathcal{P}_l : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ which equal zero if and only if the corresponding predicate is satisfied. We may also wish to optimize other objectives (*e.g.*, smoothness of a curve), encoded by energy terms $\mathcal{E}_1, \ldots, \mathcal{E}_k$. Overall, then, we have an optimization problem

$$\min_{\mathbf{p} \in \mathbb{R}^m} \sum_{i=1}^{k} \mathcal{E}_i(\mathbf{p}) \quad \text{s.t.} \quad \sum_{i=1}^{l} \mathcal{P}_i(\mathbf{p}) = 0. \tag{1}$$

In particular, we use the penalties defined in Section 3; Section 5.3 gives energy definitions in the context of various layout examples.

2 BACKGROUND

Notation. We use $|\mathbf{x}|$ and $\langle \mathbf{x}, \mathbf{y} \rangle$ to denote the Euclidean norm and inner product of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ (which are typeset in bold); $\mathbf{0}$ denotes the origin in $\mathbb{R}^n$. We also use ∂A and A^c for the boundary and complement of an n-dimensional region $A \subset \mathbb{R}^n$, *resp.*

2.1 Minkowski Difference

The *Minkowski sum* $A + B$ of any two sets $A, B \subset \mathbb{R}^n$ is the set

$$A + B := \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\}.$$

Intuitively, it describes the set obtained by "sweeping" a copy of A over every point of B and vice-versa (see Figure 3, *left*). For brevity, we will use $A + \mathbf{x}$ to denote the Minkowski sum with the set $\{\mathbf{x}\}$. The *Minkowski difference* $A - B$ is[2] the sum of A with the negated set $-B := \{-\mathbf{b} : \mathbf{b} \in B\}$. Importantly, A and B intersect if and only if $\mathbf{0} \in A - B$, *i.e.*, if there is at least one pair of points such that $\mathbf{a} = \mathbf{b}$.

2.2 Signed Distance Functions

For any n-dimensional region $A \subset \mathbb{R}^n$, let

$$d(\mathbf{x}, \partial A) := \min_{\mathbf{y} \in \partial A} |\mathbf{x} - \mathbf{y}|$$

denote the (unsigned) distance from any point $\mathbf{x}$ to the closest point on A's boundary. The *signed distance function (SDF)* for A is then

$$\phi_A(\mathbf{x}) := \begin{cases} -d(\mathbf{x}, \partial A) & \mathbf{x} \in A, \\ d(\mathbf{x}, \partial A) & \mathbf{x} \notin A. \end{cases}$$

The zero level set of ϕ_A gives the boundary of A, *i.e.*, $\partial A = \phi_A^{-1}(\mathbf{0})$ (where $^{-1}$ denotes the preimage). In our method all SDFs are expressed in closed form rather than via, *e.g.*, grids [Osher et al. 2004] or neural networks [Yariv et al. 2021, 2023; Marschner et al. 2023].

[2]Some authors instead use "Minkowski difference" to mean the set $A \ominus B = (A^c - B)^c$.

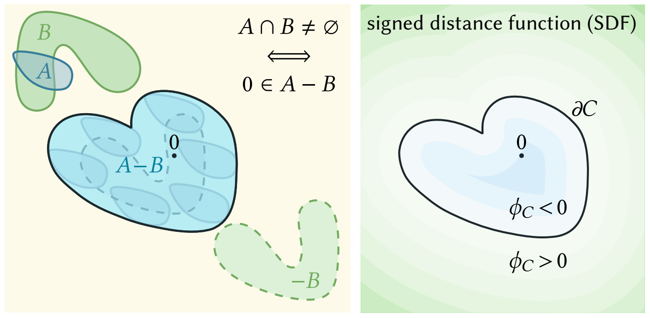


Fig. 3. Construction of the Minkowski difference $C = A - B$ between general shapes A and B; and visualization of its signed distance function ϕ_C.

3 PENALTIES

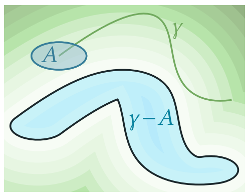
Our basic approach is to translate the logic of geometric *predicates* (*e.g.*, "A must not contain B") into differentiable *penalty functions*. In principle one could consider, *e.g.*, a complete Boolean logic—we here focus on predicates commonly found in 2D illustration.

As a starting point, let $C := A - B$ be the Minkowski difference of any two sets $A, B \subset \mathbb{R}^n$. As noted in Section 2.1, these sets intersect if and only if $\mathbf{0} \in C$—or equivalently, if $\phi_C(\mathbf{0}) < 0$. The quantity $\phi_C(\mathbf{0})$ is known in robotics as the *penetration depth*, and gives the minimum distance one would have to move A or B to resolve or induce overlap (depending on sign).

To generalize beyond overlap, we compose penetration depth with several basic functions. In particular, each predicate below is satisfied if and only if the associated nonnegative penalty evaluates to zero. Moreover, the penalty gradient is zero when the constraint is satisfied—and points toward a feasible state when it is not.

Predicate	Penalty
do not overlap ($A \cap B = \varnothing$)	$\mathcal{P}_d(A, B) = -\min(0, \phi_C(\mathbf{0}))$
overlap ($A \cap B \neq \varnothing$)	$\mathcal{P}_o(A, B) = \max(0, \phi_C(\mathbf{0}))$
tangent ($A \cap B \subset \partial A \cap \partial B \neq \varnothing$)	$\mathcal{P}_t(A, B) = \lvert\phi_C(\mathbf{0})\rvert$
contains ($A \subset B$)	$\mathcal{P}_c(A, B) = \mathcal{P}_d(A, B^c)$

(Section 3.3.1 details how to evaluate set complement for polygons.) By further composing penalties with basic functions and geometric operations, one could build even more expressive constraints. For example, ensuring mutual overlap of three sets $A \cap B \cap C \neq \varnothing$ might be enforced by evaluating $\mathcal{P}_o(A \cap B, C)$ on an explicit intersection of A and B—though we do not implement such a penalty here.



Open Curves. An open curve γ (like a line segment or hemicircle) does not have a well-defined "inside" and "outside." Hence, one cannot directly formulate penalties in terms of the SDF of γ itself (*à la* [Chen et al. 2023]). However, the Minkowski difference of γ with another set A (including another open curve) still has a well-defined inside/outside, apart from highly degenerate arrangements such as two parallel, collinear segments. Our Minkowski penalty formulation therefore applies without modification to open curves, as seen in Figure 1 (P), Figure 8 *left*, and Figure 15.

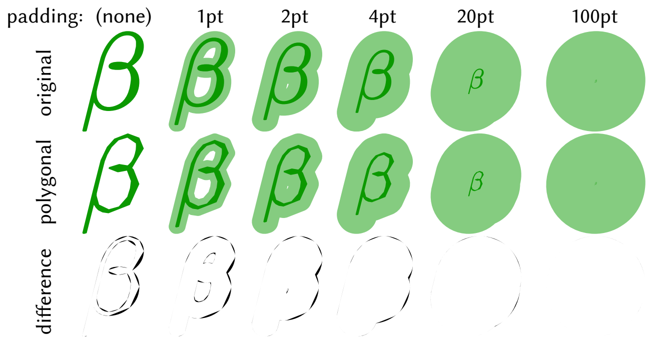
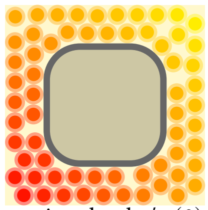


Fig. 4. When laying out vector graphics, like text labels, one often seeks padding around shapes—corresponding to a Minkowski sum with a disk. Even for small padding, little is hence gained by using the exact glyph geometry to enforce constraints (approaching identical regions in the limit).



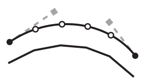
Padding. Many layout tasks demand precise *padding* between shapes, *i.e.*, a gap of constant, user-specified thickness $w \in \mathbb{R}_{>0}$. A common case is accounting for stroke width, which effectively pads the original shape. A benefit of our SDF-based formulation is that we achieve exact padding by simply adding or subtracting w from the penetration depth $\phi_C(\mathbf{0})$ (Figure 14, *right*). Otherwise, penalties are unchanged. Such offsets also provide exact constraint enforcement for rounded rectangles (inset), or more general paths with a round join style.

3.1 Evaluating Signed Distance

To use our penalties, we must be able to evaluate the penetration depth $\phi_C(\mathbf{0})$ of a Minkowski difference C, as well as the gradient $\nabla_{\mathbf{p}}\phi_C(\mathbf{0})$ with respect to the degrees of freedom $\mathbf{p}$ (from Equation 1). We here describe how to evaluate the SDF of Minkowski differences; Section 4.1 discusses gradient evaluation.

For many basic shapes A, B (*e.g.*, a circle and a rectangle), the Minkowski difference $C = A - B$ is itself a basic shape whose SDF is easily evaluated in closed form (see supplement, Appendix A). For more general shapes, we apply polygonal approximation (Section 3.2). An exact evaluation scheme for general Bézier curves would be overkill in our context: for one, early stages of optimization require only a descent direction, and not the exact gradient. Also, when padding is used, the difference between exact and approximate shapes quickly becomes negligible (Figure 4). Moreover, unlike differentiable rasterization, which needs only the distance to a single Bézier curve [Li et al. 2020], we need the distance *to the Minkowski difference*—which is far more complex [Lee et al. 1998].

3.2 Polygonal Approximation

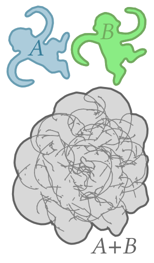


Prior to computing Minkowski differences, we approximate all Bézier curves by polygons. For our examples we simply use the endpoints of each Bézier segment plus four uniformly-spaced interior points, though one could use a more careful approximation—Li et al. [2023, Section 1] catalog several options. In particular, to strictly enforce non-overlap or containment constraints, one would need a polygon that conservatively inscribes/is circumscribed by the curve (*resp.*), *à la* Sacht et al. [2015]. For dynamic Bézier curves one could also express each polygon vertex as a linear combination of control points, though we did not implement such a scheme.

3.3 Minkowski Difference of Polygons

Given polygonal approximations of A, B, we explicitly construct their Minkowski difference $C = B - A$, which will again be a polygon. Penetration depth $\phi_C(\mathbf{0})$ is then evaluated via standard formulas for the SDF of a polygon, in linear time (see supplement, Appendix A.4).

When A and B are convex, their Minkowski difference is just the convex hull of the difference of their vertex sets—which can again be computed in linear time [Semenov 2020]. More generally, for simple nonconvex polygons there are two basic methods: *decomposition* and *convolution* [Wein et al. 2023]. Decomposition partitions A and B into convex regions, takes all pairwise Minkowski sums of these regions, and merges the sums [Margalit and Knott 1989]. However, both partitioning and polygon union can be hard to implement efficiently and robustly [Behar and Lien 2011, Section II].



We instead opt for convolution, which is efficient in practice, and comes with a well-established theory [Baram 2013]. More importantly, it enables us to differentiate through Minkowski sums (hence our penalties), as detailed in Section 4.1. Basic convolution sums all edges of A with all vertices of B, and vice-versa, yielding a superset of ∂C. It then splits segments at intersections, traces out loops, and removes "false loops" to obtain the final sum [Guibas and Seidel 1986]. *Reduced* convolution [Behar and Lien 2011] omits reflex vertices, significantly reducing the initial number of segments (see inset).

Acceleration. If the vertices of P and Q can move freely, their Minkowski sum must be recomputed at every optimization step. However, in the fairly common case where P and Q experience only rigid translations by offsets $\mathbf{p}, \mathbf{q} \in \mathbb{R}^2$, their sum can be computed just once and cached. This simplification follows from the identities

$$(P + \mathbf{p}) + (Q + \mathbf{q}) = (P + Q) + (\mathbf{p} + \mathbf{q}) \quad \text{and} \quad \phi_{C+\mathbf{c}}(\mathbf{0}) = \phi_C(-\mathbf{c}).$$

Also note that in some situations (*e.g.*, to prevent overlap of glyphs with small holes, like the letter 'B') it is sufficient just to compute the outermost boundary of the Minkowski sum and ignore removal of "false loops"—thereby skipping the slowest step of the reduced convolution algorithm.

3.3.1 More General Polygons. The strategies outlined above immediately apply to degenerate polygons—*e.g.*, a line segment can be treated as a polygon with two identical but oppositely-oriented edges. One can also compute a Minkowski sum involving a set complement A^c by just reversing the orientation of A (*i.e.*, by reversing the order of the vertices). Finally, nonsimple polygons can be partitioned into triangles [Hertel and Mehlhorn 1983] or simple polygons [Subramaniam 2003]; like the decomposition approach, one can then take the union over all pairwise sums of elements from the decomposition to obtain the Minkowski sum of the two original polygons.

4 OPTIMIZATION

We next describe a general approach for solving the vector graphics layout problem from Equation 1, using automatic differentiation (Section 4.1) and an *exterior point method* (Section 4.2).

4.1 Automatic Differentiation

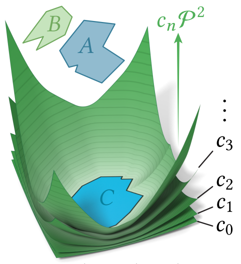
We use *reverse-mode autodiff* [Speelpenning 1980] to differentiate penalties from Section 3. Tools from machine learning, like *PyTorch* [Paszke et al. 2019], focus on relatively simple tensor-based computation and are ill-suited to our scalar- and algorithm-heavy penalty functions. Here, more conventional engines such as *Zygote* [Innes et al. 2019] and *Enzyme* [Moses and Churavy 2020] are more suitable. In particular, since we target web-based applications (Section 5.3.5), we use *Rose* [Estep et al. 2024], which runs natively in the browser (and was two orders of magnitude faster than Zygote).

To handle Minkowski sums, we could naïvely differentiate the whole convolution algorithm, but at a significant price: the adjoint pass must work through a huge number of edge calculations that are ultimately thrown out. Moreover, even though the number of arithmetic operations used by autodiff differs from function evaluation by only a constant factor, reverse-mode autodiff still asymptotically increases the number of memory operations [Jakob et al. 2022, Section 2.2]—which are orders of magnitude slower than arithmetic. We hence take a different route and track how each vertex of the sum C depends on vertices of A and B. In particular, each such vertex is either (i) a simple sum or (ii) the result of an edge-edge intersection—dramatically reducing dependency depth relative to the naïve strategy. These relationships are then provided to the autodiff engine as custom derivative expressions.

4.2 Exterior Point Method

Following a strategy suggested by Ye et al. [2020], we adopt an *exterior point method* to solve Equation 1. This method permits infeasable initialization, via progressive stiffening of constraints. Recall that our penalties $\mathcal{P}_i$ are carefully designed to equal zero when constraints are satisfied, and to be positive (hence active) only when constraints are violated. We can therefore convert Equation 1 into a sequence of unconstrained optimization problems

$$\min_{\mathbf{p}\in\mathbb{R}^m} \sum_{i=1}^{k} \mathcal{E}(\mathbf{p}) + c_n \sum_{i=1}^{l} \mathcal{P}_i^2(\mathbf{p}), \quad n = 0, 1, 2, \cdots \tag{2}$$



where positive parameters $c_0, c_1, c_2, \ldots$ progressively increase the strength of the penalties. As in barrier methods, progressively increasing c keeps successive minimizers close to each other—avoiding steep gradients that cause trouble for convergence. In particular, we follow the schedule suggested by Jensen and Bard [2002, Appendix A], letting $c_{n+1} = \eta c_n$ for $\eta = 10$, starting at $c_0 = 10^{-3}$, and stopping when the change in the objective or in $|\mathbf{p}|$ is less than $\varepsilon = 10^{-3}$. Each unconstrained problem is solved using L-BFGS [Liu and Nocedal 1989], using a line search strategy from Lewis and Overton [2009] which is guaranteed to converge to local minima even for nondifferentiable objectives.

4.3 Guarantees and Failure Modes

Convergence of our overall method to a feasible point is formally guaranteed only if one can find global minimizers for each unconstrained problem [Luenberger et al. 1984, p. 400]. However, it is easy to get some intuition for why Minkowski penalties are so robust in practice. Suppose, for instance, we want to enforce the no-overlap predicate $A \cap B = \emptyset$, via the penalty $\mathcal{P}_d(A, B)$. For simplicity, imagine that A is fixed, and B can only translate by an offset $\mathbf{p} \in \mathbb{R}^2$. Then we have a 2D energy landscape, which in an infeasible state just looks like the penetration depth $\phi_C(\mathbf{0})$ (up to sign). Since ϕ_C is an SDF, we get a nonzero gradient $|\nabla_{\mathbf{p}}\phi_C| = 1$ whenever A and B overlap.[3] As a result, *an infeasible state cannot be a fixed point of gradient descent.* Moreover, since the gradient norm is bounded away from zero (namely, always equal to 1), we cannot have an infeasible limit point. We also cannot encounter cycles, since gradient descent reduces a potential at every step (strictly enforced by line search). Of course, this rough argument may not apply to more general shape parameterizations—and in practice we can get stuck in infeasible states due to a balance of forces from different constraints (Figure 12). More rigorous analysis is an interesting question for future work.

5 RESULTS AND EVALUATION

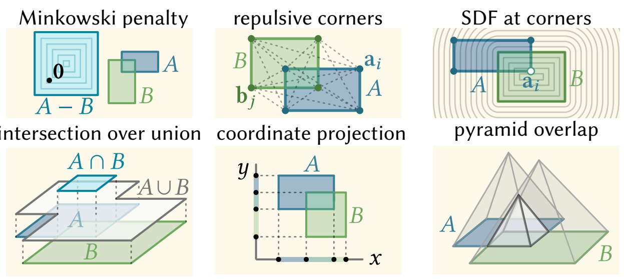
In this section we evaluate the effectiveness of our approach and compare it to alternative methods. Note that although we show only one or two images for each example, these results are not "cherry picked": as seen in Figure 15 of the supplement, we reliably obtain high-quality results for different random initializations.

5.1 Rectangle Penalties

As discussed in Section 1, simple ad-hoc penalties often fail to work as expected. As a case study, we consider several penalties specially tailored to prevent overlap of axis-aligned rectangles:

- **Intersection over union.** The overlap area $|A \cap B|$ normalized by total area $|A \cup B| = |A| + |B| - |A \cap B|$—known as *intersection over union* [Rezatofighi et al. 2019]. Here the gradient can be zero even when A and B overlap (*e.g.*, they form a "cross", or $A \subset B$).
- **Coordinate projection.** The minimum of the horizontal and vertical range of overlap, which is nonzero if $A \cap B \neq \emptyset$. Again, the gradient can be zero in, *e.g.*, nested or "cross" configurations.
- **Repulsive corners.** The sum of Coulomb potentials $1/|\mathbf{a}_i - \mathbf{b}_j|$ over all pairs of rectangle corners $\mathbf{a}_i, \mathbf{b}_j$. Here, a balance of Coulomb forces can still yield overlapping configurations.
- **SDF at corners.** The sum of $\min(0, -\phi_A(\mathbf{b}_i))$ over all corners of B and likewise for A, which is zero if all corners of A are outside B and vice versa. Here, rectangles can again overlap in a "cross."
- **Pyramid overlap.** Consider a pyramid over each rectangle, with height inversely proportional to area. Since tall, narrow pyramids "pierce through" large, flat ones, their overlap volume has a nonzero gradient for intersecting configurations [Jacobson 2021]. However, this volume is hard to differentiate (we use finite differences of mesh booleans), and does not apply to general shapes.

[3] In fact, the gradient is numerically well-defined even at nondifferentiable points, since autodiff computes *intensional derivatives*, as defined by Lee et al. [2020, Section 3].



Approach	(Min)	(Diff)	(Grad)	(Shape)	(Param)
Intersection over union	✓	✓	·	·	✓
Coordinate projection	✓	✓	·	·	·
Repulsive corners	✓	✓	·	✓	✓
SDF at corners	·	✓	·	·	✓
Pyramid overlap	✓	·	✓	·	✓
Minkowski penalty	✓	✓	✓	✓	✓

Table 1. Comparison of penalties aimed at preventing rectangle overlap—which either cannot guarantee non-overlap, or do not generalize to shapes beyond rectangles. Minkowski penalties satisfy all the target properties.

Table 1 displays a qualitative comparison of these approaches, and Figure 16 illustrates the various failure modes on a box packing problem. In general, it becomes clear that a robust penalty must consider all points of a shape (not just, *e.g.*, vertices), and must also decrease in magnitude as a configuration approaches feasibility (in order to supply useful gradients).

5.2 Comparison to Direct Distance Evaluation

One possible concern is that constructing explicit Minkowski sums is impractical relative to simpler alternatives. To gauge this cost, we compared to a simple baseline where we use a "halfplane trick" to approximate ϕ_C *without* constructing an explicit polygon for C (see supplement, Appendix B for details).

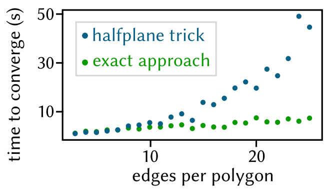


Fig. 5. Computing explicit Minkowski differences *(exact approach)* provides significant acceleration relative to direct SDF evaluation *(halfplane trick)*.

Figure 5 compares these two approaches, measuring wall-clock time (in `JavaScriptCore`) for optimizing pairwise disjoint constraints on 100 regular polygons of varying degree. Apart from fairly small polygons, we find that the overhead of computing an explicit Minkowski sum is well worth the effort. Moreover, it is unclear how to directly evaluate the Minkowski SDF for more general (*e.g.*, nonconvex) shape pairs.

5.3 Examples

Since our scheme is based on continuous optimization (versus, *e.g.*, combinatorial [Wagner et al. 2001] or spectral [Cui et al. 2023] methods), it can easily be integrated with energetic objectives that arise in a wide variety of illustration and diagramming tasks. We here consider several examples demonstrating that Minkowski penalties "play well" with common energies. Pure feasibility problems (*i.e.*,

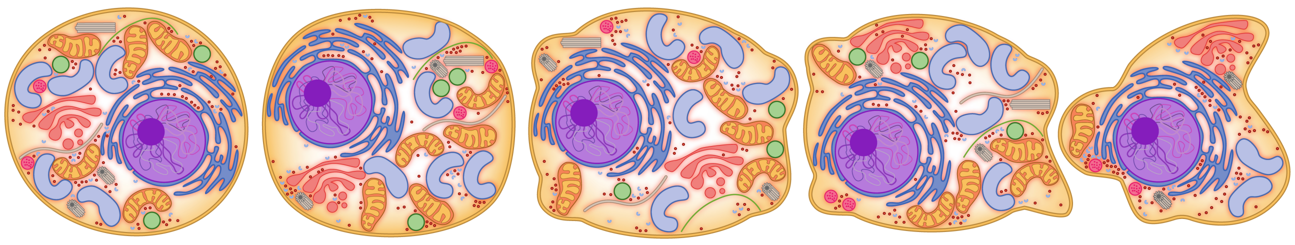


Fig. 6. To help illustrate a eukaryotic cell undergoing necrosis, we pack a user-specified collection of highly nonconvex organelle shapes into a varying outer membrane shape *(left to right)*. Individual shapes are "remixed" from existing vector art (shown in Figure 18).

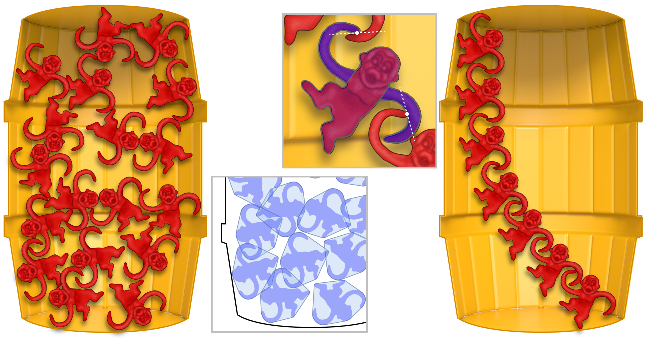


Fig. 7. For nonconvex shapes *(left)*, we get a much tighter packing than with a convex hull approximation *(center bottom)*. We also support constraints beyond no-overlap—here, tangency constraints ensure that each monkey in the barrel links arms with its neighbor *(right)*. We treat each monkey as three polygons *(center top)* to avoid, *e.g.*, tangency between heads and feet.

no energy) are shown in Figure 7 and Figure 13, demonstrating the exemplary ability of our method to handle tight packing and tangency constraints, even for highly nonconvex geometry.

The median compute time for all examples was about 3 seconds; more elaborate packings (like Figure 1 (H)) took about a minute. Table 2 in the supplement gives per-figure timings. Long optimization times were invariably due to packing together many small elements, since we naïvely evaluate all $O(n^2)$ pairs of constraints/gradients. However, since our penalties are compactly supported, this cost could be reduced to $O(1)$ by adopting, *e.g.*, an active set approach with a spatial hash (*à la* Teschner et al. [2003]).

5.3.1 Elastic Curves. *Euler elastica*, which minimize total squared curvature subject to, *e.g.*, length or endpoint constraints, provide visually pleasing arrows, connector lines, and other curved diagram elements [Levien 2009]. For optimization, we parameterize curves as polylines and minimize a discrete elastic energy (see Appendix C.1 of the supplement). Solutions are then visualized as Catmull-Rom splines [Catmull and Rom 1974]. Figure 8 and Figure 15 show examples where we simultaneously optimize the shape of elastic curves (each with 70 control points) and the arrangement of solid disks, while keeping all shapes non-intersecting and contained within a bounding region via penalties $\mathcal{P}_d$ and $\mathcal{P}_c$ (*resp.*). Notice that we find smooth curves and feasible arrangements, even in the presence of open, self-intersecting curves (which have no well-defined inside/outside), and nonconvex domains.

5.3.2 Spectral Graph Drawing. Graph drawing is a central component of data visualization, with a rich algorithmic history [Tamassia 2013]. One popular approach is *force directed layout*, which puts emphasis on edge length, but does not directly account for node shape—some methods, *e.g.*, apply node constraints heuristically after layout, or in discrete steps [Bastian et al. 2009]. In Figure 8 we augment spectral graph layout with Minkowski penalties $\mathcal{P}_d$ to explicitly avoid node-node overlap during optimization (see Appendix C.2 of the supplement for details). In turn, node shape and size can be used to clearly convey additional information not possible with traditional graph drawing algorithms.

5.3.3 Dimensionality Reduction. Similarly, we can look for a low-dimensional embedding of high-dimensional data points, while also ensuring that the shapes used to draw the points are non-overlapping. In Figure 8 for instance we embed fonts in the plane according to their typographical attributes (cap height, stroke contrast, *etc.*), and using the font name itself to represent each point. To do so, we take an approach inspired by t-SNE [van der Maaten and Hinton 2008]: minimize the *Kullback-Leibler divergence* between the similarity matrix of high-dimensional features and the current 2-dimensional embedding (see supplement, Appendix C.3 for details). We can also use Minkowski penalties to impose a prior on the topology of the low-dimensional embedding, by constraining it to a given region (Figure 8, *right*).

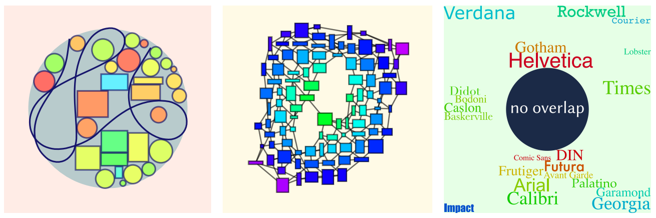


Fig. 8. Enforcing constraints via differentiable penalties enables our approach to be integrated with other energies commonly used for illustration and data visualization. Here for instance we simultaneously optimize elastic curve energy *(left)*, enabled by our ability to handle open curves, perform spectral graph layout with disjoint node representations *(center)* and show t-SNE embedding of popular fonts displayed as non-overlapping text *(right)*.

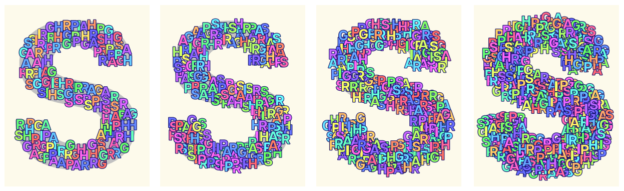


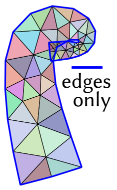
Fig. 9. Our method fails gracefully when constraints cannot be satisfied. Here, for instance, we seek to place each glyph within the larger "S", while avoiding pairwise overlap between the glyphs. Although there are far too many glyphs to achieve a feasible arrangement, the resulting layout still evenly distributes overlap across the illustration.

5.3.4 Text Layout. Typographical glyphs are most typically represented via piecewise quadratic or cubic Bézier curves, making it possible to use our framework for fine-grained placement of, *e.g.*, text labels within larger illustrations or diagrams. Figure 1 (S) provides a stress test of typographical layout, providing much tighter packing than with a simple bounding box approximation (inset). In many practical situations, diagrams become too crowded to avoid overlap altogether—here our method fails gracefully, equally distributing overlap across the layout (Figure 9).

5.3.5 Mathematical Diagrams. Penrose [Ye et al. 2020] is an open source, web-based package for mathematical diagrams [Ye et al. 2020]. For over two years, our Minkowski penalty scheme has served as the constraint enforcement strategy for shape predicates, and "battle-tested" by a community of real users. Interactive examples can be found on the Penrose gallery examples page,[4] some of which are depicted in Figure 10. Even with a client-side, browser-based *TypeScript* implementation, our method is fast enough to give immediate feedback, and robust enough to handle unanticipated layout problems generated by real users—or even randomly-generated [Harriman 2021] and LLM-generated problems [Jain et al. 2023].

5.3.6 Biological Illustration. An end-to-end example in Figure 6 demonstrates the ability of our method to handle large numbers of complex, highly nonconvex shapes. Here we "remix" a collection of organelle and membrane shapes (*resp.*) from two existing pieces of vector art (see supplement, Figure 18). Bézier curves are approximated via polygons *à la* Section 3.2; Minkowski penalties are then used to enforce containment within the cell and no-overlap between organelles. The resulting dense packing of many organelles better reflects the "crowded" nature of eukaryotic cells, which is challenging to lay out by hand (or even algorithmically [Lok 2011]).

5.3.7 Globally Injective Flattening. Though our main focus is 2D vector illustration, Minkowski penalties are also promising for broader problems in geometry processing and simulation. For example, a fundamental geometric task is to flatten a mesh to the plane without self-overlap. Here we can use Minkowski penalties $\mathcal{P}_d$ to enforce no-overlap on all pairs of non-adjacent triangles, while minimizing a standard distortion energy (see supplement, Appendix C.5 for details). Figure 11 shows an experiment inspired by the most challenging "crossing arms" example from Du et al. [2021, Figure 15, *right*] (on which that method fails). Here we reliably find globally-injective maps, starting with random, infeasible initialization, even in the presence of additional containment constraints. We conjecture that this method succeeds because (i) the weaker penalties used in the initial phase of exterior-point optimization allow us to "tunnel through" infeasible states, and (ii) we penalize full triangle-triangle overlap, rather than just, *e.g.*, edge-edge intersections along the boundary—which fail to yield injectivity (see inset). Though our current implementation prevents us from scaling to larger meshes (all $O(n^2)$ constraints are active), future extensions suggest rich new opportunities for geometry processing and beyond (Section 7).



[4]https://penrose.cs.cmu.edu/examples

6 RELATED WORK

The literature on shape optimization is vast, since this problem belongs to many communities. We here briefly survey use of Minkowski sums in various visual and geometric domains, as well as adjacent approaches that may serve as alternatives.

Robotics and Motion Planning. In robotics, penetration depth is commonly used to resolve or prevent collisions in a dynamical context [Kim et al. 2002]. For instance, Minkowski differences are used to detect interpenetration [Kockara et al. 2007], or suggest a direction for contact resolution [Dobkin et al. 1993], but we did not find work that directly differentiates penetration depth. Conversely, recent work on differentiable collision detection/resolution does not

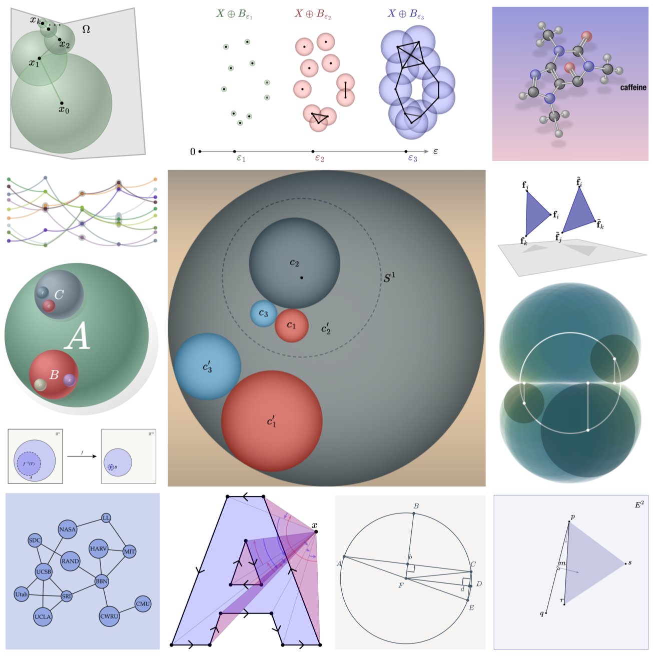


Fig. 10. Collage of diagram examples created by integrating our Minkowski-based constraint enforcement with the *Penrose* system of [Ye et al. 2020].

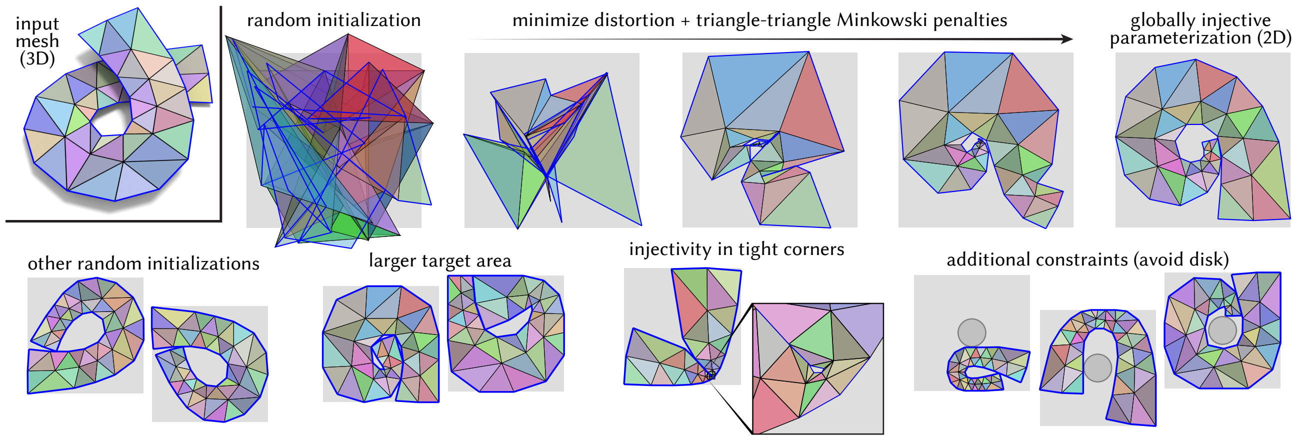

input mesh (3D)
random initialization
minimize distortion + triangle-triangle Minkowski penalties
globally injective parameterization (2D)
other random initializations
larger target area
injectivity in tight corners
additional constraints (avoid disk)

Fig. 11. Minkowski penalties also hold potential for robust geometry processing. Here for instance we compute a globally injective flattening of the 3D mesh *(top left)* into the 2D plane; since this mesh has two "crossing arms," it must bend significantly. *Top row:* even with a random initialization, we obtain a map with no self-intersection, confined to the given box. *Bottom row:* this behavior is consistent across random initializations, even with larger target areas, tight corners, and nonconvex constraints.

use Minkowski-based penalties [Zimmermann et al. 2022], and is limited to convex geometry [Tracy et al. 2023; Montaut et al. 2023]. Many Minkowski-based methods from robotics consider only simple shape parameterizations (*e.g.*, rigid motions) due to the historical difficulty of working with more complex configuration spaces [Ganjugunte 2007]. By thoughtful integration with automatic differentiation (Section 4.1) and SDF evaluation (Section 3), our approach enables the Minkowski approach to be applied to a more flexible shape parameterizations, as needed in vector illustration (and, potentially, areas like soft robotics).

Physical Simulation and Geometry Processing. Recent work from computer graphics focuses on guaranteed *global injectivity* for tasks ranging from surface parameterization to elastodynamic simulation [Fang et al. 2021; Yu et al. 2021a; Fang et al. 2021]. Since the objective is to find a non-intersecting state, most of these methods consider only feasible initialization, often using domain-specific knowledge [Smith and Schaefer 2015]. These methods also consider only volumetric domains that have an inside/outside [Bridson et al. 2005]—using, *e.g.*, the closest point "outside" an object to drive collision resolution [Chen et al. 2023]. In contrast, we cannot rely on problem-specific initialization strategies, since constraints are often user-defined and hard to anticipate (Section 5.3.5). Moreover, we must handle rich constraints beyond no-overlap, and open curves that do not have a well-defined inside/outside. Though some strategies in this space address infeasible states, they are highly-specialized to a certain problem domain such as injective mapping [Du et al. 2021; Garanzha et al. 2021] or mesh repair [Fargion and Weber 2022], and are not easily adapted to our problem. Minkowski-based methods also appear in collision detection for real-time graphics[Montaut et al. 2022]—with largely the same considerations as in robotics.

Design and Illustration. Constraint-based design has a long history in computer graphics, dating back to Sutherland [1963]. Some systems focus mainly on equality-type constraints, enforced via a nonlinear solver [Nelson 1985]. *Dynamic geometry* packages such as Cinderella [Richter-Gebert et al. 2012] or Geogebra [Hohenwarter and Hohenwarter 2002] handle complex geometric relationships, but in a fundamentally different way: using a parameterized model where parameters can be adjusted bidirectionally [Kortenkamp 1999] (akin to forward/inverse kinematics). This approach is not directly useful for inequality-type constraints (like containment or overlap), but it could be quite interesting to integrate a kinematic model with our energetic penalties. Some work uses repulsive potentials to generate collision-free 2D graphics [Saputra et al. 2018; Yu et al. 2021b], but considers only the no-overlap condition, and requires feasible initialization. Li et al. [2020] and [Vicini et al. 2022] differentiate vector graphics and SDFs, but do not formulate shape layout penalties. Element-based texture synthesis [Ma et al. 2011] yields impressive packing of small repetitive shapes, but relies on fairly uniform sampling and shape size to achieve efficient overlap removal [Hsu et al. 2020]. Finally, Ye et al. [2020] proposed a system (Penrose) for constraint-based layout of vector graphics, using coarse approximations (*e.g.*, bounding boxes) that do not take precise geometry into account. As shown in Section 5.3.5, our approach can be used to enhance the capabilities of existing design and illustration systems, such as Penrose.

Data Visualization and Cartography. In dataviz, the number of primitives tends to be quite large (*e.g.*, a whole social network); graphical primitives are often reduced to, *e.g.*, simple boxes for layout [Sun et al. 2023], and existing work focuses more on reducing clutter and improving readability [Gibson et al. 2013]. In contrast, vector illustration demands high-quality placement of a small number of objects (10s–100s), attending carefully to their specific geometry. Some dataviz layout techniques use Minkowski differences—but share limitations with methods from robotics, such as supporting only no-overlap constraints; like methods from physical simulation they also assume a well-defined inside/outside. For instance, Dwyer

[2009] uses the *minimum penetration depth* vector to adjust node-link visualizations, but allows only convex polygons, and does not consider differentiable optimization.

Cartography considers more fine-grained label placement [Imhof 1975]. Most methods for automatic label placement are algorithmic and gradient-free (*e.g.*, simulated annealing) [Christensen et al. 1992; Bekos et al. 2019; Kern and Brewer 2008]. van Kreveld and Schlechter [2005] use Minkowski sums for overlaps between label boxes and island polygons, but also assume a definite inside/outside and consider only no-overlap constraints.

7 LIMITATIONS AND FUTURE WORK

Limitations of our optimization scheme are detailed in Section 4.3. As noted in Section 5.3, the biggest performance bottleneck is in "all-pairs" examples, where our naïve $O(n^2)$ evaluation scheme could easily be replaced with an $O(1)$ active set approach (*e.g.*, using spatial hashing). It might also be natural to adopt a coarse-to-fine strategy, *e.g.*, start with bounding boxes, and use polygonal approximation only near the end of optimization. Piecewise differentiable penalties could in principle cause problems for some numerical schemes—though we have carefully chosen optimization and differentiation strategies that work well here (Sections 4.2, 4.1). Complex constraint sets yield large computational graphs, which can in turn cause slow code generation—a challenge most directly addressed by further work on autodiff systems (rather than our penalty formulation). Moreover, as noted in Section 5.3.5, our method already provides usable interaction for real users on fairly complex diagrams (Figure 10). In general we are optimistic that the efficiency, robustness, and flexibility of our framework makes it a natural candidate for integration into broader tools for diagramming and vector illustration.

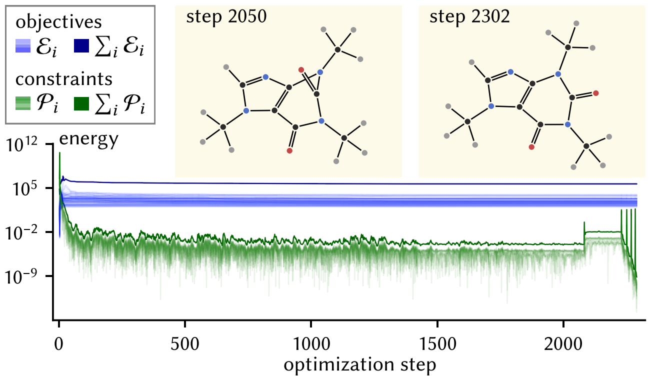


Fig. 12. In general our method may of course get stuck in local minima, due to a balance of forces between different penalties and objective terms. Here for instance we obtain an infeasible state during an early stage of the exterior point method *(top center)*, though in this case, stiffening the penalty kicks optimization into a feasible state *(top right)*.

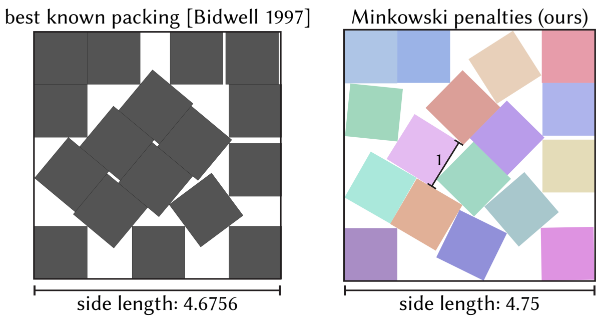


Fig. 13. Although we make no guarantees about global optimality, our Minkowski scheme generally finds excellent layouts—even for problems it was not specifically designed to handle. Here we achieve a similar solution to the best known packing of 17 unit squares into the smallest possible box [Friedman 2012].

REFERENCES

Alon Baram. 2013. *Polygonal Minkowski Sums Via Convolution: Theory and Practice.* University of Tel-Aviv.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks. *Proceedings of the International AAAI Conference on Web and Social Media* 3, 1 (Mar. 2009), 361–362. https://doi.org/10.1609/icwsm.v3i1.13937

Evan Behar and Jyh-Ming Lien. 2011. Fast and robust 2D Minkowski sum using reduced convolution. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 1573–1578. https://doi.org/10.1109/IROS.2011.6094482

Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg. 2019. External Labeling Techniques: A Taxonomy and Survey. *Comput. Graph. Forum* 38, 3 (jun 2019), 833–860.

Robert Bridson, Sebastian Marino, and Ronald Fedkiw. 2005. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses.* 3–es.

Edwin Catmull and Raphael Rom. 1974. A class of local interpolating splines. In *Computer Aided Geometric Design.* Academic Press, 317–326. https://doi.org/10.1016/B978-0-12-079050-0.50020-5

He Chen, Elie Diaz, and Cem Yuksel. 2023. Shortest Path to Boundary for Self-Intersecting Meshes. *arXiv preprint arXiv:2305.09778* (2023).

Jon Christensen, Joe Marks, and Stuart Merrill Shieber. 1992. Labeling point features on maps and diagrams. (1992).

Blake Courter. 2023. *Unit Gradient Fields: SDFs, UGFs, and Their Friends.* https://www.blakecourter.com/2023/05/18/field-notation.html

Keenan Crane, Fernando De Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses.* 1–126.

Keenan Crane and Max Wardetzky. 2017. A glimpse into discrete differential geometry. *Notices of the American Mathematical Society* 64, 10 (2017).

Qiaodong Cui, Victor Rong, Desai Chen, and Wojciech Matusik. 2023. Dense, Interlocking-Free and Scalable Spectral Packing of Generic 3D Objects. *ACM Trans. Graph* 42, 4 (2023).

David Dobkin, John Hershberger, David Kirkpatrick, and Subhash Suri. 1993. Computing the intersection-depth of polyhedra. *Algorithmica* 9, 6 (1993), 518–533.

Xingyi Du, Danny M Kaufman, Qingnan Zhou, Shahar Z Kovalsky, Yajie Yan, Noam Aigerman, and Tao Ju. 2021. Optimizing global injectivity for constrained parameterization. *ACM Trans. Graph.* 40, 6 (2021), 260–1.

Tim Dwyer. 2009. Scalable, Versatile and Simple Constrained Graph Layout. *Comput. Graph. Forum* 28, 3 (jun 2009), 991–998.

Sam Estep, Raven Rothkopf, Wode Ni, and Joshua Sunshine. 2024. Rose: Efficient and Extensible Autodiff on the Web. arXiv:2402.17743 [cs.PL]

Yu Fang, Minchen Li, Chenfanfu Jiang, and Danny M Kaufman. 2021. Guaranteed globally injective 3D deformation processing. *ACM Transactions on Graphics* 40, 4 (2021).

Guy Fargion and Ofir Weber. 2022. Globally Injective Flattening via a Reduced Harmonic Subspace. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–17.

Erich Friedman. 2012. Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics* (2012), DS7–Aug.

Shashidhara K Ganjugunte. 2007. A Survey on Techniques for Computing Penetration Depth. (2007).

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. 2021. Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–16.

Helen Gibson, Joe Faith, and Paul Vickers. 2013. A survey of two-dimensional graph layout techniques for information visualisation. *Inf. Vis.* 12, 3-4 (jul 2013), 324–357.

Leonidas Guibas and Raimund Seidel. 1986. Computing convolutions by reciprocal search. In *Proceedings of the second annual symposium on Computational geometry.* 90–99.

Hwei-Shin Harriman. 2021. Edgeworth: authoring diagrammatic math problems using program mutation. In *Companion Proceedings of the 2021 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*. 22–24.

Stefan Hertel and Kurt Mehlhorn. 1983. Fast triangulation of simple polygons. In *Foundations of Computation Theory*, Marek Karpinski (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 207–218.

Markus Hohenwarter and Markus Hohenwarter. 2002. GeoGebra. *Available on-line at http://www. geogebra. org/cms/en* (2002).

Chen-Yuan Hsu, Li-Yi Wei, Lihua You, and Jian Jun Zhang. 2020. Autocomplete element fields. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

Nathan Hurst, Wilmot Li, and Kim Marriott. 2009. Review of automatic document formatting. In *Proceedings of the 9th ACM symposium on Document engineering*. 99–108.

Eduard Imhof. 1975. Positioning names on maps. *Am. Cartogr.* 2, 2 (jan 1975), 128–144.

Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B Shah, and Will Tebbutt. 2019. A Differentiable Programming System to Bridge Machine Learning and Scientific Computing. arXiv:1907.07587 [cs.PL]

Alec Jacobson. 2021. personal communication.

Rijul Jain, Wode Ni, and Joshua Sunshine. 2023. Generating Domain-Specific Programs for Diagram Authoring with Large Language Models. In *Companion Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*. 70–71.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. Dr. jit: A just-in-time compiler for differentiable rendering. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–19.

Paul A Jensen and Jonathan F Bard. 2002. *Operations research models and methods*. John Wiley & Sons.

Jill Phelps Kern and Cynthia A. Brewer. 2008. Automation and the map label placement problem: A comparison of two GIS implementations of label placement. *Cartogr. Perspect.* 60 (2008), 22–45.

Young J Kim, Miguel A Otaduy, Ming C Lin, and Dinesh Manocha. 2002. Fast penetration depth computation for physically-based animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 23–31.

Sinan Kockara, Tansel Halic, Kamran Iqbal, Coskun Bayrak, and Richard Rowe. 2007. Collision detection: A survey. In *2007 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 4046–4051.

Yehuda Koren. 2003. On spectral graph drawing. In *International Computing and Combinatorics Conference*. Springer, 496–508.

Ulrich H Kortenkamp. 1999. *Foundations of dynamic geometry*. Ph. D. Dissertation. ETH Zurich.

In-Kwon Lee, Myung-Soo Kim, and Gershon Elber. 1998. Polynomial/rational approximation of Minkowski sum boundary curves. *Graphical Models and Image Processing* 60, 2 (1998), 136–165.

Wonyeol Lee, Hangyeol Yu, Xavier Rival, and Hongseok Yang. 2020. On correctness of automatic differentiation for non-differentiable functions. *Advances in Neural Information Processing Systems* 33 (2020), 6719–6730.

Raphael Linus Levien. 2009. *From spiral to spline: Optimal techniques in interactive curve design*. University of California, Berkeley.

Adrian Lewis and Michael L. Overton. 2009. Nonsmooth Optimization via BFGS. *SIAM Journal on Optimization*, 1–35.

Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. 2020. Differentiable Vector Graphics Rasterization for Editing and Learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020), 193:1–193:15.

Yajuan Li, Meng Zhang, Wenbiao Jin, and Chongyang Deng. 2023. Approximating Bézier curves with least square polygons. *The Visual Computer* (2023), 1–10.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1 (1989), 503–528.

Corie Lok. 2011. Biomedical illustration: From monsters to molecules. *Nature* 477, 7364 (2011), 359–361.

David G Luenberger, Yinyu Ye, et al. 1984. *Linear and nonlinear programming*. Vol. 2. Springer.

Chongyang Ma, Li-Yi Wei, and Xin Tong. 2011. Discrete element textures. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–10.

Avraham Margalit and Gary D Knott. 1989. An algorithm for computing the union, intersection or difference of two polygons. *Computers & Graphics* 13, 2 (1989), 167–183.

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 121, 12 pages. https://doi.org/10.1145/3610548.3618170

Louis Montaut, Quentin Le Lidec, Antoine Bambade, Vladimir Petrik, Josef Sivic, and Justin Carpentier. 2023. Differentiable collision detection: a randomized smoothing approach. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3240–3246.

Louis Montaut, Quentin Le Lidec, Vladimir Petrik, Josef Sivic, and Justin Carpentier. 2022. Collision Detection Accelerated: An Optimization Perspective. In *Robotics: Science and Systems*.

William Moses and Valentin Churavy. 2020. Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12472–12485. https://proceedings.neurips.cc/paper/2020/file/9332c513ef44b682e9347822c2e457ac-Paper.pdf

Greg Nelson. 1985. Juno, a constraint-based graphics system. In *Proceedings of the 12th annual conference on Computer Graphics and Interactive Techniques*. 235–243.

Stanley Osher, Ronald Fedkiw, and K Piechor. 2004. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.* 57, 3 (2004), B15–B15.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Inigo Quilez. 2020. *The SDF of a Line Segment*. YouTube. https://www.youtube.com/watch?v=PMltMdi1Wzg

I. Quilez. 2021. *2D distance functions*. https://iquilezles.org/www/articles/distfunctions2d/distfunctions2d.htm

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. 2019. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 658–666. https://doi.org/10.1109/CVPR.2019.00075

Jürgen Richter-Gebert, Ulrich H Kortenkamp, Jürgen Richter-Gebert, and Ulrich H Kortenkamp. 2012. Interactive Geometry with Cinderella. *The Cinderella. 2 Manual: Working with The Interactive Geometry Software* (2012), 67–152.

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested cages. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–14.

Reza Adhitya Saputra, Craig S Kaplan, and Paul Asente. 2018. RepulsionPak: Deformation-driven element packing with repulsion forces. In *Proceedings of the 44th Graphics Interface Conference*. 10–17.

Yury Semenov. 2020. Minkowski sum of convex polygons - Algorithms for Competitive Programming. https://cp-algorithms.com/geometry/minkowski.html

Jason Smith and Scott Schaefer. 2015. Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.

Bert Speelpenning. 1980. *Compiling fast partial derivatives of functions given by algorithms*. University of Illinois at Urbana-Champaign.

Lavanya Subramaniam. 2003. *Partition of a Non-Simple Polygon into Simple Polygons*. Master's thesis. University of South Alabama, Mobile, Alabama.

Guodao Sun, Zihao Zhu, Gefei Zhang, Chaoqing Xu, Yunchao Wang, Sujia Zhu, Baofeng Chang, and Ronghua Liang. 2023. Application of Mathematical Optimization in Data Visualization and Visual Analytics: A Survey. *IEEE Trans. Big Data* 9, 4 (aug 2023), 1018–1037.

Ivan E Sutherland. 1963. Sketchpad: A man-machine graphical communication system. In *Proceedings of the May 21-23, 1963, spring joint computer conference*. 329–346.

Roberto Tamassia. 2013. *Handbook of graph drawing and visualization*. CRC press.

Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. 2003. Optimized spatial hashing for collision detection of deformable objects.. In *Vmv*, Vol. 3. 47–54.

Kevin Tracy, Taylor A Howell, and Zachary Manchester. 2023. Differentiable collision detection for a set of convex primitives. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3663–3670.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. https://api.semanticscholar.org/CorpusID:5855042

Marc van Kreveld and Tim Schlechter. 2005. Automated label placement for groups of islands. In *Proc. of the 22nd International Cartographic Conference*.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022), 125:1–125:18. https://doi.org/10.1145/3528223.3530139

Frank Wagner, Alexander Wolff, Vikas Kapoor, and Tycho Strijk. 2001. Three rules suffice for good label placement. *Algorithmica* 30 (2001), 334–349.

Ron Wein, Alon Baram, Efi Fogel, Eyal Flato, Michael Hemmer, and Sebastian Morr. 2023. CGAL 5.6 - 2D Minkowski Sums: User Manual. https://doc.cgal.org/5.6/Minkowski_sum_2/index.html

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. 2023. BakedSDF: Meshing Neural

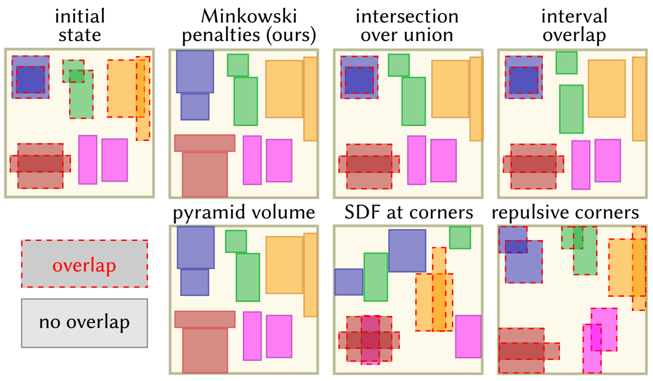


Fig. 16. Even for the simple case of rectangles, most ad-hoc schemes cannot eliminate overlap. The lone exception is pyramid overlap, which (unlike our Minkowski penalties) does not generalize to shapes beyond rectangles.

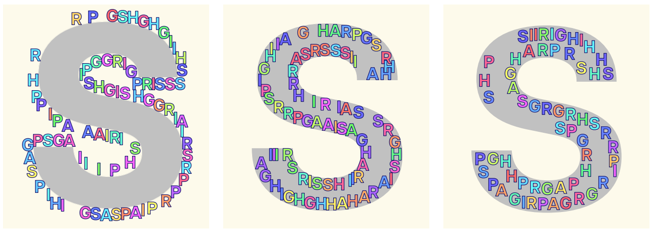


Fig. 14. Results for alternative geometric queries for the layout of letters interacting with the large “S”. The individual glyphs can be tangent to the “S” shape *(left)* or its complement (center) while remaining disjoint with each other. The padding of both tangent and disjoint queries can also be set to nonzero values *(right)*.

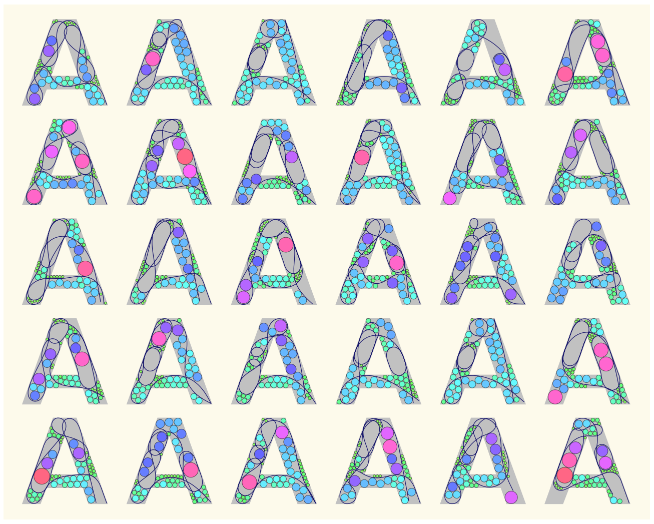


Fig. 15. Starting from different random seeds provides many design alternatives for a given set of constraints/objectives. Here we show the result of 30 different random initializations for the example described in Section 5.3.1.

SDFs for Real-Time View Synthesis. *arXiv* (2023).

Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Jonathan Sunshine, and Keenan Crane. 2020. Penrose: From Mathematical Notation to Beautiful Diagrams. *ACM Trans. Graph.* 39, 4 (2020).

Chris Yu, Caleb Brakensiek, Henrik Schumacher, and Keenan Crane. 2021a. Repulsive Surfaces. *ACM Trans. Graph.* 40, 6 (2021).

Chris Yu, Henrik Schumacher, and Keenan Crane. 2021b. Repulsive curves. *ACM Transactions on Graphics (TOG)* 40, 2 (2021), 1–21.

Simon Zimmermann, Matthias Busenhart, Simon Huber, Roi Poranne, and Stelian Coros. 2022. Differentiable collision avoidance using collision primitives. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8086–8093.