The Bounding Simplex: A SIMD-Era Rejection Primitive

Bryan McNett
bryan@mcnett.org

Abstract

We present the *bounding simplex*, a rejection primitive that bounds an object with $d+1$ half-spaces in d dimensions where the axis-aligned bounding box (AABB) uses $2d$. A query supplies the opposing $d+1$ half-spaces, so rejection costs $d+1$ comparisons instead of $2d$ — four against six in 3D — and a single simplex already encloses a finite volume, so it can reject on its own. The paper makes three contributions, in sequence. First, we establish the simplex as a primitive: its looser fit is inconsequential for sparse content, and although it is a net liability on scalar hardware it becomes an advantage past a crossover SIMD width — which is why, we argue, so elementary a bound went unbuilt for so long. Second, we show that the projection axes $\{X, Y, -(X+Y)\}$, rather than equal-angle directions, are preferable, for reasons of numerical stability and alignment with axis-aligned content. Third, we show that the intersection of two opposing simplices on these axes is nothing more than an AABB with two opposite corners cut off — a shape that unites the simplex's fast one-sided rejection with the box's tightness, and that is geometrically a $(2d{+}2)$-DOP never previously recognized as the intersection of simplices.

1 Introduction

A world holds a large *spatial database* of objects, and we must repeatedly answer one question: given a *query* object — a moving body, a ray, a selection volume — which stored objects could it intersect? Testing every object exactly is far too slow, so a *broad phase* first discards those that *cannot possibly* intersect the query, leaving a short candidate list for the expensive exact test. That discard is a conservative *rejection test*: each object is wrapped in a simple bounding volume, and if its bound provably cannot overlap the query's, the object is rejected at once. Because this test is evaluated across the whole database and repeated for every query, its per-test cost is what governs the broad phase — and making it cheaper is the subject of this paper.

The axis-aligned bounding box (AABB) is the near-universal primitive for this test, bounding an object with $2d$ half-spaces — two per axis. We argue that $2d$ is more than necessary: a *bounding simplex* of only $d+1$ half-spaces suffices to enclose a finite volume and reject a query, and on wide-SIMD hardware it does so faster.

A theme organizes the paper: the simplex is a consequence of *wide* SIMD, not merely an optimization for it. The simplex is a net liability on scalar hardware — its looser fit costs more than its fewer planes save, because per-object early-out makes plane count nearly free — and its advantage emerges only past a crossover SIMD width, where early-out collapses and plane count dominates. We argue this is why the idea lay unexplored *as a bounding volume* for so long: the hardware that rewards it did not yet exist.

A word on naming. The essential primitive is the *simplex* — the triangle in 2D, the tetrahedron in 3D. Its dual, the intersection with an opposing simplex, we call the *AABO* (axis-aligned bounding

octahedron); Part III shows it is nothing but an AABB with two corners cut off. "AABO" is a deliberately catchy label and a concession to readers accustomed to the box, but the simplex is the idea and the octahedron is the courtesy.

Contributions and structure. The paper is in three parts.

1. **The simplex** (Section 2). We establish the bounding simplex as a rejection primitive — $d+1$ half-spaces, finite volume, one-sided rejection — show that its looser fit is inconsequential for sparse content, and argue that wide SIMD is what turns it from liability into advantage. We situate it against the bounding box and the sphere.

2. **The axes** (Section 3). We show, for the first time, that the projection axes $\{X, Y, -(X+Y)\}$ are preferable to equal-angle directions, for reasons of numerical stability and alignment with axis-aligned content.

3. **The intersection of opposing simplices** (Section 4). We show, for the first time, that the intersection of two opposing simplices on these axes is exactly an AABB with two opposite corners cut off, uniting the simplex's fast rejection with the box's tightness — and that this shape, though geometrically a $(2d{+}2)$-DOP, has never been recognized as the intersection of simplices.

2 Part I: The Bounding Simplex

2.1 The Box and the Sphere

Before weighing them, Figure 1 introduces the two axis sets we use — the rectangular X, Y and the three simplex axes A, B, C spaced 120° apart — and Figure 2 fixes what each primitive *is* in 2D and how many scalars it stores, shown beneath each shape as a $1 \times N$ array of cells. The box pins two parallel faces per axis along X and Y — $\min x,\ \max x,\ \min y,\ \max y$, four scalars $(2d)$. The bounding circle stores a center and a radius — $x,\ y,\ r$, three scalars $(d{+}1)$. The bounding simplex stores one face per axis along three axes A, B, C that sum to zero — $\min a,\ \min b,\ \min c$, three scalars $(d{+}1)$. The axes here are drawn at equal 120° angles, the symmetric choice; Part II (Section 3) argues for the asymmetric $\{X, Y, -(X{+}Y)\}$ instead. The simplex thus matches the circle's compactness while keeping the box's flat, one-scalar-per-facet faces — with one fewer face than the box.

The axis-aligned bounding box is old enough to predate any single attribution — it appears across early graphics, CAD, and computational geometry as the bounding volume one reaches for without deliberation, in wide use at least since early ray-tracing space subdivision [1]. It is the intersection of $2d$ half-spaces in d dimensions. Tellingly, the count $2d$ was never derived as optimal; the box was adopted because it is obvious and cheap, not because anyone showed two half-spaces per axis to be the right number. A primitive that wins by obviousness is rarely re-examined, which is part of why the question this paper asks — whether $d+1$ faces suffice — went unasked for so long.

Bounding spheres [7] are as compact as the simplex — $d+1$ scalars, a center and a radius — but their rejection structure is the opposite. A sphere is a single quadratic constraint, not an intersection of half-spaces, so it offers no facet whose violation can be read from one scalar. Its exact rejection reads all $d+1$ scalars and forms a squared distance (multiply–adds, not comparisons), with no early-out within the test. Even the cheapest conservative test — separation along a single axis — must read the radius together with a center coordinate, because the radius couples into

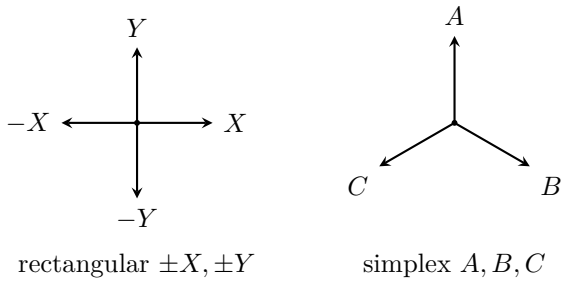


Figure 1: The two axis sets. Left: the rectangular axes, which the box uses as *four* directions $\pm X, \pm Y$ — two antipodal pairs, one $[\min, \max]$ slab each ($2d$=4). Right: the three simplex axes A, B, C, spaced 120° apart, summing to zero, and *unpaired* (d+1=3). The pairing into opposing slabs is exactly what the simplex drops, trading one direction's worth of planes for a looser fit. Drawn symmetric here; Part II replaces A, B, C with $\{X, Y, -(X+Y)\}$.

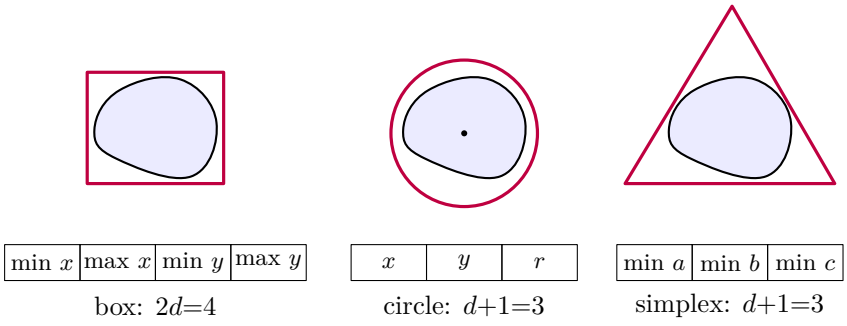
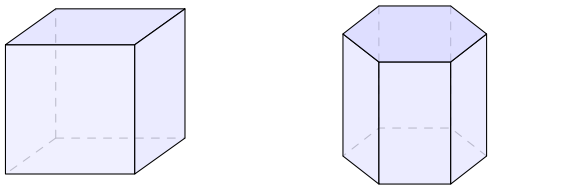


Figure 2: The three primitives bounding the same object, and — beneath each — the scalars it stores as a $1 \times N$ array. The box needs four (two extents per axis on X, Y); the circle three (center x, y and radius r); the simplex three (one min per axis on A, B, C, Figure 1). The box spends $2d$ values where the circle and simplex spend d+1; the simplex keeps the box's flat, one-scalar-per-facet faces with one fewer face. Axes at equal 120° angles; Part II replaces them with $\{X, Y, -(X+Y)\}$.

every axis; there is no analogue of a box face or a simplex facet that rejects on one value alone. On SIMD hardware this costs twice over: a compare is cheaper than a multiply–add, and a compare yields a lane bitmask directly. The simplex occupies a position neither competitor reaches: it keeps the sphere's $d+1$-scalar compactness while retaining the box's decomposable, compare-only, one-scalar-per-facet rejection — with fewer facets than the box ($d+1$ versus $2d$).

Background: the k-DOP. Several bounds discussed here — the box among them — are *discrete oriented polytopes*, and Part III turns on the term, so we fix it now. A k-DOP is a convex polytope bounded by k half-spaces whose outward normals are drawn from a small, *fixed* set of directions used in antipodal pairs: each pair $(+a, -a)$ is a slab $[\min, \max]$ along one direction a, and the polytope is the intersection of those slabs (so k is even). The AABB is the smallest case — the three coordinate directions alone, a 6-DOP in 3D (a 4-DOP in 2D). Adding further fixed directions tightens the fit at the cost of more planes: a cube is the 6-DOP of the coordinate axes, while a hexagonal prism is an 8-DOP — a hexagonal cross-section (three in-plane directions 60° apart) closed by two end caps, eight faces in four antipodal pairs (Figure 3). The two commitments that define the family — directions *shared* across all objects, and planes in *parallel opposing pairs* — are exactly the ones the bounding simplex will break.



cube (6-DOP) hexagonal prism (8-DOP)

Figure 3: Two discrete oriented polytopes in 3D. **Left:** a cube — the 6-DOP of the three coordinate directions (three slab pairs). **Right:** a hexagonal prism — an 8-DOP: a hexagonal cross-section (three in-plane directions 60° apart) closed by two end caps, eight faces in four antipodal pairs. The AABB is the simplest k-DOP; adding fixed directions buys a tighter fit at more planes.

Comparisons among bounding volumes are well established. Cost-function evaluations of bounding-volume effectiveness date to early ray tracing [9]; collision-detection surveys catalogue the sphere, box, k-DOP, and oriented-box tradeoffs [11]; and the standard references tabulate them systematically [10, 12]. These treat each volume as a monolith, weighed on tightness, storage, and test cost. What none of them do — and what this paper turns on — is decompose the box's half-space set, ask whether *fewer* faces suffice, or analyze the per-scalar rejection cost that, under wide SIMD, decides the comparison (Section 2.5).

The simplex is not new as a *shape* to enclose. The geometry of the smallest enclosing simplex is decades old: O'Rourke et al. give an optimal algorithm for the minimum-area enclosing triangle in 2D [4], and Zhou and Suri do the same for the minimum-volume enclosing tetrahedron in 3D [5]. Both are pure computational geometry; neither applies the simplex as a runtime rejection primitive, which is the gap this paper fills. Two near relatives must also be set aside. The simplex of the GJK algorithm [6] lives in the Minkowski difference as a witness that the origin is contained — it encloses nothing in world space and is not a conservative proxy. And although a bounding tetrahedron superficially resembles a "4-DOP," it is not a k-DOP [2]: k-DOPs use a *shared, fixed* direction set with *paired antipodal* slabs, and that pairing is exactly what enables their interval-overlap test. Our simplex breaks both assumptions — per-object orientation, and $d+1$ *unpaired*, non-parallel planes.

Zachmann observed that the DOP machinery generalizes to unpaired directions [3], yet the field only ever instantiated the slab form; the $d+1$-plane simplex went unbuilt.

One domain in graphics did adopt the simplex over the hypercube, and on a cost argument structurally like ours. Perlin's simplex noise [8] replaced the cubic lattice of classic gradient noise — which interpolates among the 2^n corners of a hypercell — with a simplicial lattice whose cells have only $n+1$ corners, expressly because the 2^n growth made the original intractable in four and higher dimensions. The parallel is worth stating because it is, to our knowledge, the only place the field has preferred the simplex to the hypercube for scaling reasons — but it did so for *interpolation*, where the saving is over cell corners (2^n versus $n+1$, exponential), and never carried the instinct to *rejection*, where the saving is over facets ($2d$ versus $d+1$, and merely linear). The bounding simplex is that same preference applied to the operation it was never applied to; that the facet saving is only linear, not exponential, is likely part of why it drew no attention.

2.2 Definition

In d dimensions, choose $d+1$ projection axes $a_0, \ldots, a_d$ such that no hyperplane through the origin leaves all positive-axis points in a single open hemisphere. A practical choice — the axes $\{X, Y, -(X+Y)\}$ examined in Part II (Section 3) — is:

$$\begin{aligned} a_0 &= X \\ a_1 &= Y \\ a_2 &= Z \\ &\vdots \\ a_d &= -(X + Y + Z + \cdots) \end{aligned}$$

The up-simplex S^+ is defined by minimum projections along these axes:

$$S^+ = \{\, p \mid \min_0 \le \mathrm{proj}(a_0, p),\ \ldots,\ \min_d \le \mathrm{proj}(a_d, p) \,\}.$$

The down-simplex S^- is the dual:

$$S^- = \{\, p \mid \mathrm{proj}(a_0, p) \le \max_0,\ \ldots,\ \mathrm{proj}(a_d, p) \le \max_d \,\}.$$

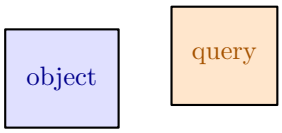
Each simplex independently encloses a finite volume, and the hemisphere condition is exactly what secures it — the reason the count is $d{+}1$ rather than d. A basis of d axes *spans* $\mathbb{R}^d$: every point is some combination of them. But its positive hull is only a cone, so d directions *reach* every point while *enclosing* none; surrounding the origin takes at least $d{+}1$. That extra direction is the difference between coordinatizing space and containing it. It is also the critical difference from the AABB, whose axis-directions are exactly such a basis: its d "min" half-spaces bound only an unbounded orthant (in 2D, any 3 of its 4 half-spaces still leave an infinite region), so no subset of the box's half-spaces can reject a query on its own. A simplex's $d{+}1$ surrounding directions can.

2.3 The Rejection Test

Given a query bounded by S_q^- and a world object bounded by S_w^+, rejection requires $d{+}1$ comparisons:

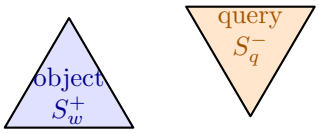
$$\mathrm{rejects}(q, w) = (w.\min_0 > q.\max_0) \ \vee\ (w.\min_1 > q.\max_1) \ \vee\ \cdots$$

This is $d+1$ comparisons versus AABB's $2d$ (Figure 4). For $d=3$: 4 vs 6; for $d=4$: 5 vs 8. The advantage grows with dimension.




```
intersect =
  object.minx <= query.maxx &&
  object.maxx >= query.minx &&
  object.miny <= query.maxy &&
  object.maxy >= query.miny
```

(a) AABB: a paired min / max test per axis, $2d$=4 comparisons.




```
intersect =
  object.mina <= query.maxa &&
  object.minb <= query.maxb &&
  object.minc <= query.maxc
```

(b) Simplex: one-sided test per axis, d+1=3 comparisons. Object is an up-simplex, query a down-simplex.

Figure 4: Rejection as interval tests on the candidate axes; in each panel the two shapes are drawn disjoint. **(a)** Two AABBs intersect only if their projections overlap on *both* coordinate axes — a paired min / max test per axis, $2d$=4 comparisons. **(b)** An object up-simplex S_w^+ (storing three minima) against a query down-simplex S_q^- (storing three maxima): they intersect only if $w.\min \le q.\max$ on each of the three axes A, B, C, so d+1=3 comparisons suffice and any single failing axis rejects. The simplex test never reads a paired min / max on one axis — only the object's minima against the query's maxima, the opposing half-spaces.

If the simplex test fails to reject, the opposing simplex may be loaded:

$$\text{rejects}(q, w) = \text{simplex\_reject}(q.\text{down}, w.\text{up}) \ \vee \ \text{simplex\_reject}(q.\text{up}, w.\text{down}).$$

The second test rarely executes; objects that pass the first simplex test and the second are in proximity and genuinely require full intersection testing.

2.4 The Looseness Is Not a Problem

A recurring objection is that the simplex, having fewer half-spaces, is a looser fit than the box — a tetrahedron circumscribes more empty volume than a cube. This is true, and it is the right thing to worry about: looseness produces false positives, which flow to the expensive exact test. The question is whether the saved half-space evaluations outweigh the extra false positives. We make the tradeoff precise using the expected-cost framework standard in bounding-volume analysis, which dates to the cost-function selection of bounding volumes for ray tracing [9] and the surface-area heuristic [13, 14].

Expected evaluations under early-out. Evaluate the m half-spaces in a fixed order, and let S_k be the event that a query survives the first k (is not yet rejected), with S_0 certain. A query reaches plane k+1 only if S_k held, so the evaluation count C satisfies $C > k \iff S_k$, and therefore

$$\mathbb{E}[C] \ = \ \sum_{k=0}^{m-1} \Pr[S_k],$$

where $\Pr[S_k]$ is the measure of the partial intersection of the first k inside-half-spaces against the query distribution, decreasing in k. The box pays terms $k = 0, \ldots, 5$; the simplex pays only $k = 0, \ldots, 3$. The two omitted terms $\Pr[S_4] + \Pr[S_5]$ are deep-prefix survivals — the *small* ones — so the early-out savings alone are modest. The decisive advantages lie elsewhere.

The SIMD regime: a deterministic reduction. Under wide SIMD, per-object early-out is ineffective: every lane evaluates every plane for the batch. The cost is then simply m compares and m scalar loads per object, so the simplex yields a flat $4/6$ — a 33% reduction in both compare instructions and bound bandwidth — independent of geometry. This structural reduction is realized only once SIMD is wide enough to defeat early-out, the subject of Section 2.5.

The cost of looseness. The false positives that survive a bound flow to the exact test. The expected number of such survivors for a query is the *survival fraction*

$$\kappa \;=\; N \cdot \frac{\mu(B)}{\mu(\Omega)},$$

where $\mu(B)$ is the configuration-space measure of the bound (its Minkowski sum with the query region — the set of query placements that survive) and $\mu(\Omega)$ is the domain measure. Looseness is the constant ratio $c = \mu(B_{\text{simplex}})/\mu(B_{\text{box}}) > 1$, *independent of N and of scene density.* The extra work it costs is $(c-1)\,\kappa_{\text{box}}$ exact tests.

Crossover. Combining the per-object cost with the survival cost gives, per query,

$$\begin{aligned}\text{Cost}_{\text{simplex}} &\approx N \cdot 4 \cdot w_{\text{cmp}} + c\,\kappa_{\text{box}} \cdot w_{\text{full}},\\ \text{Cost}_{\text{AABB}} &\approx N \cdot 6 \cdot w_{\text{cmp}} + \kappa_{\text{box}} \cdot w_{\text{full}},\end{aligned}$$

so the simplex is cheaper exactly when

$$\frac{\kappa_{\text{box}}}{N} \;<\; \frac{2\,w_{\text{cmp}}}{(c-1)\,w_{\text{full}}}.$$

Because $\kappa_{\text{box}}/N = \mu(B_{\text{box}})/\mu(\Omega) \to 0$ as object size shrinks relative to the domain, a sufficiently sparse scene always satisfies the inequality — regardless of how large c is. This is the precise form of the empirical observation that, for small objects fairly distributed in a large space, the looseness of the simplex is inconsequential: the looseness factor c is a constant multiplying a vanishing quantity. The inequality also identifies when the box wins: dense scenes, large objects, large c, or an expensive exact test w_{full}.

The right measure of looseness. The measure μ depends on the query type. For overlap and point/region queries it is volume; for ray queries it is surface area, by the Surface Area Heuristic [13, 14]. The factor c is therefore not a single constant but a property of construction and query type, and it can be large: the simplex from $\{X, Y, Z, -(X+Y+Z)\}$ circumscribing a unit cube has volume $3^3/6 = 4.5$ versus the cube's 1. Fitting the simplex to the object's own geometry rather than its AABB reduces c sharply (the axis choice of Section 3 reaches parity). We therefore treat c and κ/N as quantities to be *measured* (Section 5), not assumed.

2.5 Why Wide SIMD Favors the Simplex

The simplex's advantage is not merely enhanced by wide SIMD — it is *created* by it. On scalar and narrow-vector hardware the simplex is a net liability, and understanding why resolves the historical puzzle: how a bound this elementary could go unbuilt for so long.

Structure of Arrays (SoA) layouts [16] arrange fields for SIMD consumption by keeping all $\min_X$ values contiguous, all $\min_Y$ values contiguous, and so on. Under wide SIMD the classic "check one axis first, early-out if all lanes reject" strategy fails, because the probability of a lane intersecting grows with lane width [17]: for width w and per-slab intersection probability p, the early-out probability is p^w, which vanishes for realistic p and $w > 4$.

Early-out and plane count are substitutes. With per-object early-out, a query bails on the first rejecting half-space, so the expected evaluation count $\mathbb{E}[C] = \sum_{k=0}^{m-1} \Pr[S_k]$ (Section 2.4) is dominated by its first one or two terms for *both* shapes; plane three is rarely reached, plane six almost never. Whether $m = 4$ or $m = 6$ is therefore nearly invisible. The simplex's structural saving is hidden, while its looseness penalty $(c-1)\kappa\, w_{\text{full}}$ is paid in full. The net advantage at width one is

$$\text{Adv}(w{=}1) \;\approx\; \underbrace{0}_{\text{savings hidden by early-out}} \;-\; \underbrace{(c-1)\,\kappa\, w_{\text{full}}}_{\text{looseness, paid in full}} \;<\; 0.$$

On scalar hardware, AABB wins.

Wide SIMD removes the substitute. A SIMD instruction processes all w lanes at the same cost whether one lane is live or all w are: there is no saving from having fewer active lanes. The only way to avoid work is to skip an entire instruction — to early-out a whole batch — and that requires *every* lane to reject. A single surviving lane forces the compare to run. Per-object early-out therefore collapses under wide SIMD: the probability that all w lanes reject on a single plane is p^w (Figure 5), which vanishes for realistic per-plane batch-reject probability p and growing w. Once early-out is gone, the batch pays the full, deterministic m compares and loads, so plane count becomes the dominant cost rather than a hidden one:

$$\text{Adv}(w) \;\approx\; \underbrace{2\,N\, w_{\text{cmp}}\,(1-p^{\,w})}_{\text{plane savings, grows with } w} \;-\; \underbrace{(c-1)\,\kappa\, w_{\text{full}}}_{\text{looseness, independent of } w}\,.$$

The first term rises from ≈ 0 toward its full value $2Nw_{\text{cmp}}$ as $p^w \to 0$; the penalty is fixed in w. There is therefore a crossover width $w^\star$ below which the simplex loses and above which it wins by up to the full 4/6 (33%) reduction in compares and bound bandwidth. A threshold is not incidental — the model requires one. For p in the realistic 0.4–0.6 range, p^w is still appreciable at $w = 4$ but negligible by $w = 8$–16, placing $w^\star$ in that band.

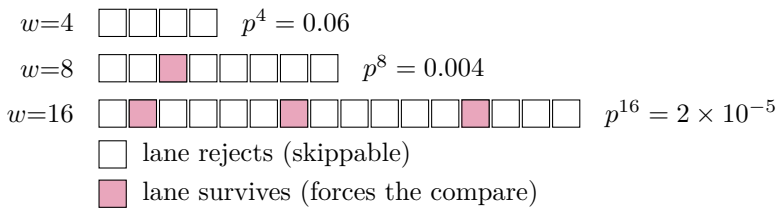


Figure 5: A SIMD compare costs the same whether one lane or all w are live, so the only way to save work is to skip the entire batch — which requires *every* lane to reject. The probability of that is p^w (shown for $p = 0.5$): a single surviving lane (shaded) forces the compare, and survivors become near-certain as w grows. Batch-wide early-out is thus a casualty of wide SIMD, which is why plane count, not early-out, governs cost there.

A bound whose time arrived with the hardware. This reframes the contribution. Width $w = 8$ is AVX (256-bit, eight floats, c. 2011); $w = 16$ is AVX-512 (c. 2016); GPU warps and wavefronts (32 / 64) are the regime in which early-out provably fails [17]. The simplex was not an overlooked trick: on a scalar machine, on early SSE (width four), or on any hardware where early-out made AABB's two extra planes effectively free, the simplex's looser fit was a pure liability. It becomes advantageous only once SIMD is wide enough to kill early-out — hardware that was

not mainstream when the AABB (1984) [1] and the BVH canon were designed. The bound is co-designed with an era of hardware rather than missed within an earlier one. (Wide SIMT on GPUs predates wide CPU vectors by a few years, so the window opens with mainstream wide vector hardware around 2011, not with any single instruction set; and the exact location of $w^\star$ depends on the measurable parameters p, c, and w_{full}, so we treat it as something to be demonstrated empirically (Section 5) rather than derived.)

3 Part II: Choosing the Axes

The simplex of Part I needs $d+1$ projection axes summing to zero, but says nothing about *which*. The symmetric choice — $d+1$ directions spaced at equal angles — is the obvious one, and is what a reader first imagines. We argue that the asymmetric, axis-aligned choice $\{X, Y, -(X+Y)\}$ is better in practice, for two separable reasons: numerical stability and alignment with the workload. To our knowledge the choice between these axis sets has not previously been examined.

The axis set $\{X, Y, -(X+Y)\}$ in 2D (and $\{X, Y, Z, -(X+Y+Z)\}$ in 3D) is not ad hoc: it is the gradient frame of the *barycentric coordinates* of the standard simplex. For the standard 2-simplex with vertices $(0,0), (1,0), (0,1)$, the barycentric functions are $\lambda_1 = x$, $\lambda_2 = y$, $\lambda_0 = 1 - x - y$, whose gradients are exactly $(1,0), (0,1), (-1,-1)$. The defining "$\sum_i \text{proj}(a_i, p) = 0$" property of the frame is the signature of barycentric coordinates. The same structure — $d+1$ vectors in d-space summing to zero — is the A_d root system in Lie theory and, in its symmetric realization, the Miller–Bravais hexagonal axes of crystallography.

It is instructive to contrast these *standard-simplex* axes with the *regular-simplex* axes (Figure 6): $d+1$ vectors spaced symmetrically (in 2D, three unit vectors 120° apart). Both are barycentric gradient frames; they differ only in the shape of the reference simplex — standard (right) versus regular (equilateral) — and are therefore *affinely equivalent*, related by the single linear shear that maps one simplex onto the other. There is no combinatorial or topological difference between them. The only difference is that the shear is *anisotropic*, and that anisotropy is the entire source of their differing behavior.

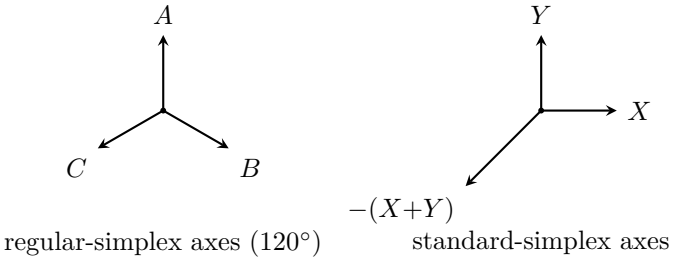


Figure 6: The two axis sets of Part II, both barycentric gradient frames summing to zero. Left: the *regular-simplex* axes — three unit directions 120° apart (the A, B, C of Part I). Right: the *standard-simplex* axes $\{X, Y, -(X+Y)\}$, the gradients $(1,0), (0,1), (-1,-1)$ of barycentric coordinates: X and Y are the coordinate axes and the third is their negated sum (length $\sqrt{2}$, at 225°). The two are affinely equivalent — related by the anisotropic shear taking the equilateral reference triangle to the right one — and that anisotropy is the whole of their differing behavior.

The standard-simplex axes have two practical advantages, which are separable. The first is construction: the X and Y projections are the identity — free and exact — and the third is a single add and negate, $-(x+y)$, also exact, whereas the regular-simplex axes require dot products with

irrational constants ($\sqrt{3}/2$ in 2D), costing multiplies and introducing floating-point error that must be absorbed by an ε-pad, slightly loosening the effective bound. Built this way the simplex reuses the same vertex extents an AABB already computes, while its rejection takes $d+1$ comparisons.

The second advantage is alignment with the workload, and it is a separating-axis (SAT) argument. The $d+1$ axes are a fixed candidate set of separating directions; a bound rejects a query only when one axis is close enough to the true separating direction to certify separation. The rejection power is therefore

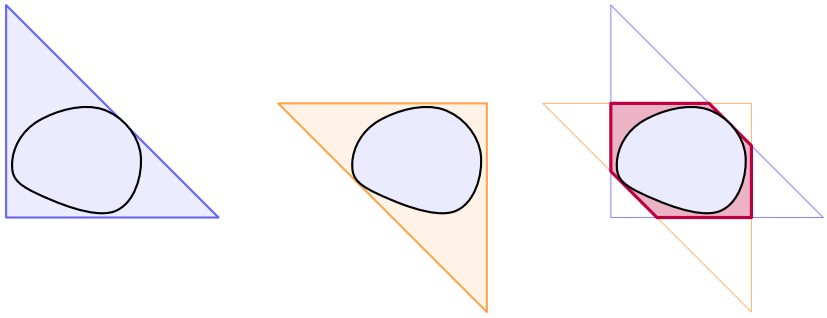
$$\Pr[\text{reject}] \;=\; \int_{S^{d-1}} \mathbf{1}[\text{some } a_i \text{ certifies separation along } u]\; d\mu(u),$$

where μ is the distribution of separating directions in the workload. Maximizing this over axis choices depends entirely on μ: for *isotropic* μ symmetry makes the regular-simplex axes optimal, but real spatial data is strongly axis-aligned (scenes authored on world axes, objects stored as AABBs, grids and screen space), so μ concentrates near the coordinate axes and the axis-aligned barycentric frame places its facets where the probability mass lies. The regular-simplex axes are optimal only under an isotropy assumption the workload violates — the same reason axis-aligned boxes routinely outperform isotropically "optimal" oriented bounds on authored data [15]. The apparent bias of the standard-simplex axes is thus not a defect overcome in practice but an advantage that matches the data; we quantify the split between the construction and alignment channels empirically in Section 5.

4 Part III: The Intersection of Opposing Simplices

4.1 An AABB with Two Corners Cut Off

Two opposing simplices — an up-simplex and a down-simplex on the same axes — always intersect in a bounded polygon (Figure 7); this section identifies exactly which one.



(a) up-simplex S^+ (b) down-simplex S^- (c) $S^+ \cap S^-$ (AABO)

Figure 7: The AABO as the intersection of two opposing simplices, on the standard-simplex axes $\{X, Y, -(X{+}Y)\}$ where each simplex is a *right* triangle. **(a)** The up-simplex $S^+ = \{x \geq a,\ y \geq b,\ x{+}y \leq D\}$ encloses the object with its right angle at the lower-left. **(b)** The down-simplex $S^- = \{x \leq a',\ y \leq b',\ x{+}y \geq E\}$ encloses it with its right angle at the upper-right. **(c)** Their intersection is the AABB $[a, a'] \times [b, b']$ with its two opposite corners sliced off by the diagonals (the faint triangles show the two simplices) — the AABO. Because the simplices are right triangles rather than equilateral, the result is a box-with-corners-cut, not the regular hexagram the symmetric axes would produce.

The up-simplex S^+ bounds an object from below, the down-simplex S^- from above (Section 2.2).

On the standard-simplex axes of Part II,

$$S^+ = \{\, x \geq a,\ y \geq b,\ z \geq c,\ x+y+z \leq D \,\}, \qquad S^- = \{\, x \leq a',\ y \leq b',\ z \leq c',\ x+y+z \geq E \,\},$$

and their intersection is

$$\underbrace{[a,a'] \times [b,b'] \times [c,c']}_{\text{the AABB}} \cap \{\, E \leq x+y+z \leq D \,\},$$

exactly the axis-aligned bounding box with its two extreme corners — the $+++$ and $---$ ends of the main diagonal — sliced off by the diagonal half-spaces. In 2D this is a rectangle with two opposite corners cut, a hexagon; in 3D, a box with two corners cut, an eight-faced solid (Figures 8 and 9). We call it an "octahedron" only in the eight-faces sense: it inherits the box's aspect ratio and can be long and thin or tall and narrow, unlike the regular Platonic octahedron. (The decomposition into two opposing tetrahedra is itself old: the regular case is the shape at the core of Kepler's stella octangula, and in crystallography the combination of the "positive" and "negative" tetrahedra of the tetrahedral hemihedry. Our construction is the axis-aligned, irregular version, which those symmetric forms do not produce.) The shape has no standard dimension-general name, so we name it by construction: the AABO.

This is the synthesis the paper builds toward. The bounding box is tight and axis-aligned but spends $2d$ half-spaces and has no subset that rejects: any d of its faces leave an unbounded region. The single simplex gives the opposite — one-sided rejection in $d+1$ comparisons (Part I) — at the price of a looser fit. The AABO has both at once. It *is* the box, hence at least as tight (Section 4.3); yet it is built from two simplices, so a query loads one simplex ($d+1$ values), rejects most candidates outright, and consults the opposing simplex only on the rare survivor. The fast one-sided rejection of the simplex and the tightness of the box, in a single primitive.

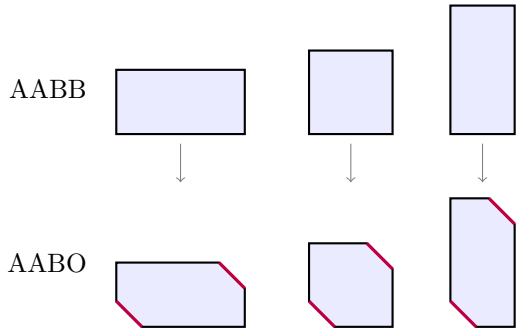


Figure 8: Top: axis-aligned bounding boxes of three aspect ratios. Bottom: the corresponding AABOs — the *same* boxes, with their two opposite (extreme) corners cut off (purple edges). The AABO inherits the box's aspect ratio exactly. The 3D shape is Figure 9.

4.2 Is It Just a k-DOP?

The shape admits the usual classifications: it is a convex polytope, a polyhedron in 3D, and — since its $2(d+1)$ planes come in $d+1$ antipodal orientations — a $(2d+2)$-DOP, a discrete oriented polytope. These are all general descriptions. "k-DOP" in particular names *any* k half-spaces in opposing pairs and is far broader than the object at hand; the specific $(2d+2)$-DOP carved by the axes $\{X, Y, Z, -(X+Y+Z)\}$ has, to our knowledge, no documented status of its own beyond having $2(d+1)$ planes. It has never been singled out or named.

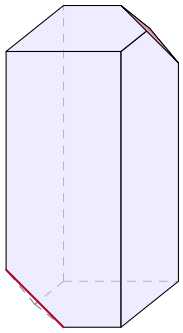
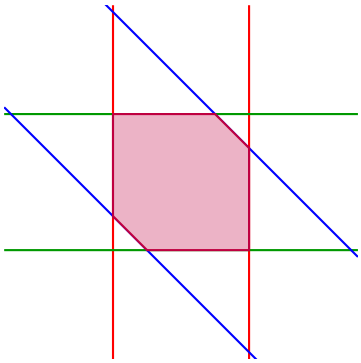


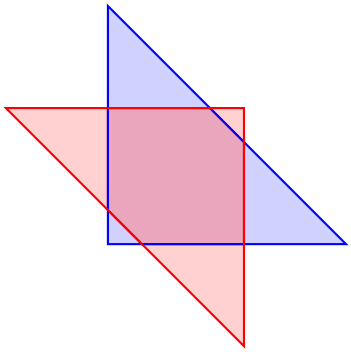
Figure 9: The 3D AABO of Section 4.1, shown tall and narrow: an axis-aligned box with two opposite corners sliced off by the diagonal half-spaces — six clipped box faces plus two triangular cuts (shaded), eight faces in all. The far (bottom) corner's cut faces away and is drawn dashed.

What the classification hides is the structure that makes it useful. A k-DOP is normally read as $k/2$ opposing pairs, one $[\min, \max]$ slab per orientation [2]. A slab is infinite off its own axis, so no pair — and no proper subset of pairs — bounds a finite volume or rejects a query; this is why k-DOP implementations load all k values per test (Figure 10a).

The same $2(d+1)$ planes regroup by *sign*, however. The $d+1$ planes whose normals face one way form the up-simplex S^+; the $d+1$ facing the other form S^- (Figure 10b). Because those $d+1$ orientations positively surround the origin, each group closes into a finite simplex on its own: with axes $\{X, Y, Z, -(X+Y+Z)\}$ the four "min" planes give $x \geq a,\ y \geq b,\ z \geq c,\ x+y+z \leq D$, a bounded tetrahedron, where the four corresponding slab ends bound nothing. The shape is the intersection of these two opposing simplices, and a single one of them already rejects.



(a) k-DOP: three opposing *slab* pairs, one per axis (X red, Y green, the diagonal blue). Each slab is infinite; no pair or subset bounds a finite region.



(b) AABO: two *simplex* sets. S^+ (blue, 3 min half-spaces) and S^- (red, 3 max half-spaces) each bound a finite triangle on their own.

Figure 10: The same six half-spaces in 2D, grouped two ways. The bounded region (purple) is identical, but the decompositions are not interchangeable: the k-DOP groups by axis into infinite slabs (a), while the AABO groups by sign into two finite, independently-rejecting simplices (b).

This regrouping — a k-DOP read as the intersection of two opposing simplices, one of which suffices to reject — is what we have not found in the k-DOP or collision-detection literature, which treats a k-DOP's planes as inseparable opposing pairs, loaded together. The two-tetrahedra decomposition is old as geometry (Section 4.1); what is new is bringing it to bounding volumes and rejecting with a single simplex.

The regrouping is a privilege of these axes, not a property of k-DOPs in general: the AABO is a *proper* subclass. A sign-group closes into a simplex only because the $d+1$ axes positively span $\mathbb{R}^d$ — no hyperplane through the origin holds them all in one closed hemisphere (Section 2.2). A hexagonal prism is the counterexample. Its eight planes are a bona fide 8-DOP — the *same* plane budget as the 3D AABO — and it bounds a finite volume, yet its six side-normals are coplanar, all perpendicular to the extrusion axis, so no choice of one plane per opposing pair positively spans space. The four "min" faces leave the extrusion axis unbounded and close only once the opposing cap is added; neither sign-group is a simplex, and neither rejects on its own. A prism is thus a k-DOP that cannot be an AABO, and the positive-spanning axes are precisely the line between the two. No prior work needed to warn against the prism: reading all k planes together makes positive spanning irrelevant, so the condition becomes a design rule only once a single sign-group is asked to reject — an operation the k-DOP literature never performs.

4.3 Is the Dual Necessary?

It is worth stating plainly what the opposing simplex is for. The single up-simplex is the performance contribution: the cheap, lane-friendly rejector whose advantage grows with SIMD width (Section 2.5). The dual — testing both simplices, equivalently bounding by the full octahedron — exists for a narrower reason. With the standard-simplex axes the octahedron is precisely the six axis-aligned AABB planes plus two diagonal corner-cuts, hence a bound *at least as conservative as the AABB* (and tighter wherever the diagonals bite). This is what it offers a practitioner migrating from an AABB pipeline: a guarantee that the replacement bound is never looser than the one it supersedes.

That guarantee is a convenience, not a necessity. The only thing the dual buys is tightness — fewer false positives forwarded to the exact test — and as parallelism grows the cost of a rejection test falls toward zero, so tightness buys less and less. In the limit the single simplex suffices: its looseness factor c (Section 2.4) multiplies a survival fraction that the cheap, wide test renders affordable. For isotropic content, the dual is a refinement that the hardware trend slowly retires.

There is one genuine exception: content with a truly axis-aligned component — architecture, cityscapes, voxel worlds, screen-space layouts. The single up-simplex retains only three of the six axis-aligned planes (the minima) and substitutes a single diagonal for the three maxima; against an axis-aligned box that substitution is loose exactly where the geometry is tight. The dual restores the full axis-aligned plane set — the embedded AABB — which is the genuinely tight bound for such geometry. This is the separating-axis argument of Section 3 again: when the workload concentrates separating directions on the coordinate axes, the axis-aligned half-spaces earn their keep and the dual is no longer optional. We therefore present the dual simplex not as a second contribution but as the AABB-compatibility mode of the first — inessential for isotropic content under wide SIMD, and load-bearing for axis-aligned content.

5 Experimental Results

We test the central prediction of Section 2.5 directly: that batch-wide early-out collapses as SIMD width w grows, driving the AABB toward its full plane count while the simplex retains its early-out, so that a crossover width $w^\star$ separates a regime where the AABB wins from one where the simplex wins.

Method. The benchmark (`simd_width_sweep.cpp`) is deliberately *platform-neutral*: it contains no SIMD intrinsics, and the per-lane inner loops are written so the compiler auto-vectorizes them

	avg. half-spaces evaluated per box		
SIMD width w	AABB	simplex	AABO
1	1.56	1.87	1.87
8	2.29	3.09	3.09
16	2.64	3.39	3.40
64	3.58	3.79	3.80
256	4.27	3.93	4.00
1024	5.07	3.98	4.23
4096	**5.77**	**4.00**	**4.90**
accepts (any w)	21 814	47 559	20 039
looseness vs. AABB	1.00	2.18	**0.92**

Table 1: As SIMD width grows, the AABB's early-out collapses and its work climbs toward the full 6 half-spaces, while the bare simplex stays at 4 and the AABO tracks it. Accept counts are independent of w and measure looseness: the bare simplex admits 2.18× as many boxes as the AABB (its looseness factor c), whereas the AABO — two opposing tetrahedra — is actually *tighter* than the AABB at 0.92×.

to the host's native width. The parameter w is the *early-out granularity* — a batch of w boxes is tested one half-space at a time, and the batch may skip the remaining half-spaces only when *all* w lanes have already rejected. This faithfully simulates a w-wide machine's gated early-out even for w far wider than the host's physical vector registers, letting us sweep $w \in \{1, 2, \ldots, 4096\}$ on commodity hardware (Apple arm64, `-O3 -ffast-math`). Crucially, *both* shapes are granted a batch early-out after *every* half-space: the AABB is given every chance to exit early, so the asymmetry that emerges is purely geometric, not an artifact of a handicapped baseline.

The scene follows the small-objects-in-a-large-world regime that motivates the bound: $N = 4.2 \times 10^6$ radius-1 objects scattered uniformly in a ± 50 cube (per-axis slab-overlap $\approx 4\%$), queried by 128 probes. We compare three shapes built on the standard-simplex axes $A{=}X$, $B{=}Y$, $C{=}Z$, $D{=}-(X{+}Y{+}Z)$: the **AABB** (6 half-spaces), the bare **bounding simplex** (4 half-spaces, the up-tetrahedron alone), and the **AABO** (8 half-spaces, the two opposing tetrahedra with the closed simplex tested first). We report a hardware-independent work metric — the average number of half-spaces evaluated per box — alongside measured wall-clock time, and the accept count of each shape (its looseness).

Early-out collapses for the AABB. Table 1 and Figure 11 show the mechanism in its purest form. At $w = 1$ the AABB evaluates only 1.56 of its 6 half-spaces per box — early-out is highly effective, and the simplex's structural saving is invisible, exactly as the width-one analysis predicts. As w grows the AABB's work climbs monotonically toward its full 6 (5.77 at w=4096), because the all-lanes-reject condition needed to skip a plane becomes vanishingly rare. The bare simplex instead converges to exactly 4.000: it closes a volume at its fourth half-space, so its early-out never dies. The AABO converges toward 4 as well (4.90 at w=4096), tracking the simplex rather than the AABB despite carrying eight planes, because its closed simplex rejects the overwhelming majority of boxes before the dual tetrahedron is ever read.

The crossover $w^\star$. Figure 12 reports measured wall-clock time per box. At $w = 1$ the AABB is fastest (2.31 vs. 3.28 ns), confirming that on scalar hardware the simplex is a net liability — its looseness is paid in full while its plane saving is hidden by early-out. The curves cross at $w^\star \approx 8$:

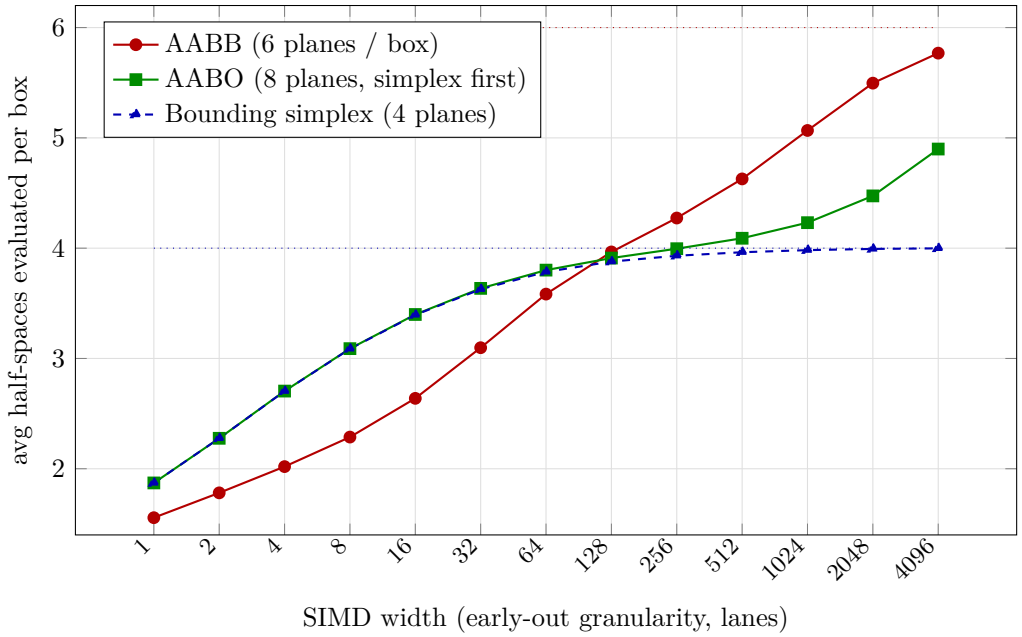


Figure 11: Average half-spaces evaluated per box vs. SIMD width (log scale). The AABB (red) rises toward 6 as batch-wide early-out fails; the bare simplex (blue) holds at 4 because four half-spaces already close a volume; the AABO (green) tracks the simplex. The work-metric curves cross near $w \approx 128$, but because that metric ignores memory traffic it understates the simplex's advantage — see Figure 12.

by $w = 16$ the AABO is 34% faster (0.314 vs. 0.478 ns) and the gap persists to the widest batches. This places $w^\star$ squarely in the $w = 8$–16 band predicted in Section 2.5, i.e. at the width of AVX and AVX-512 and well within the 32/64 of GPU warps and wavefronts. The time crossover precedes the work crossover because time also rewards the simplex's smaller bound bandwidth (four loaded values rather than six), an effect the pure half-space count does not capture.

Looseness and the dual. The accept counts in Table 1 quantify the looseness factor c of Section 2.4 independently of width. The bare simplex admits $47\,559/21\,814 = 2.18\times$ as many boxes as the AABB — the price of its single-tetrahedron looseness, paid as extra work passed downstream. The AABO, being the intersection of two opposing tetrahedra, is *tighter* than the AABB ($20\,039/21\,814 = 0.92\times$) while still early-outing after four planes. This is the paper's central trade made concrete: the AABO is simultaneously cheaper than the AABB at wide SIMD *and* a strictly tighter bound, whereas the bare simplex buys its even-lower cost with looseness. Which to prefer depends on whether the downstream narrow-phase or the bound test itself dominates.

Reproducibility. The benchmark is a single self-contained file with no dependencies beyond a C++17 compiler; the figures and tables above are regenerated from its CSV output by `plot_simd_width.py`. The crossover location depends on object size relative to the world (which sets the per-slab probability p); the optional fourth argument sweeps it.

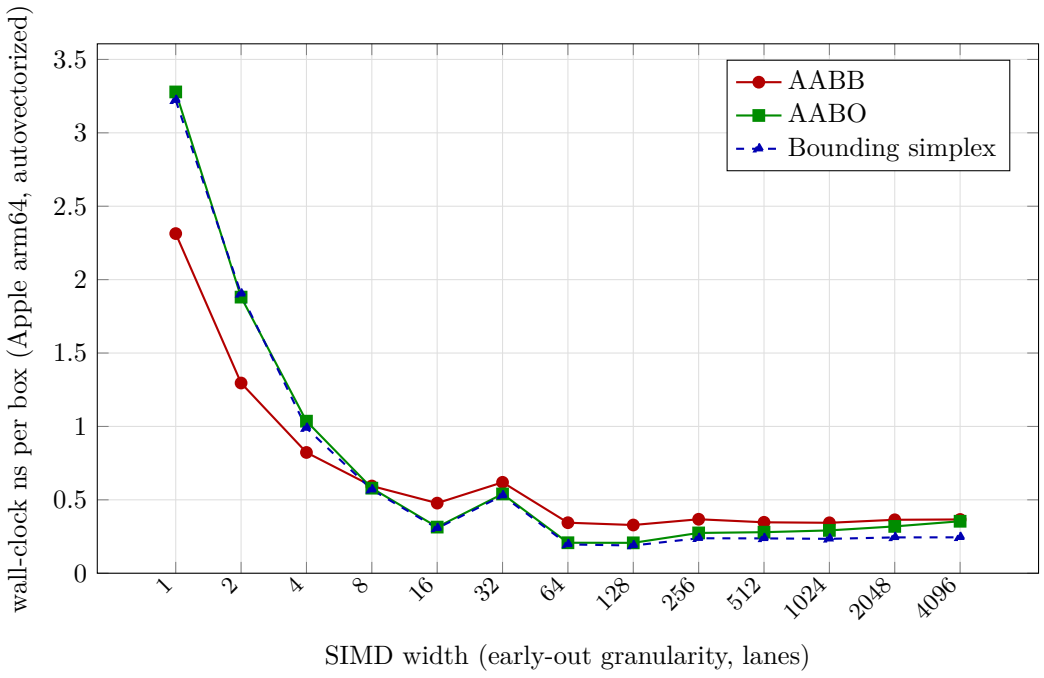


Figure 12: Measured wall-clock time per box vs. SIMD width (Apple arm64, autovectorized, no intrinsics). The AABB leads only at the narrowest widths; the simplex and AABO cross it at $w^\star \approx 8$ and lead thereafter, matching the model's prediction that the crossover falls in the 8–16 band.

6 Conclusion

We have presented the bounding simplex as a rejection primitive: $d+1$ half-spaces in place of the AABB's $2d$, with a query supplying the opposing $d+1$, and a single simplex sufficing to enclose a finite volume and reject on its own. An expected-cost model makes precise when the simplex's looser fit is inconsequential and when it is not, and a hardware argument explains why the primitive is a liability on scalar machines and a win only past a crossover SIMD width — the reason so simple a bound went unbuilt for so long. We then showed that the projection axes $\{X, Y, -(X+Y)\}$ are preferable to equal-angle directions, for numerical stability and alignment with axis-aligned content; and finally that the intersection of two opposing simplices on those axes is nothing more than an AABB with two corners cut off — an ordinary $(2d{+}2)$-DOP, but one no prior work has recognized as the intersection of two opposing simplices. The shape unites the simplex's fast one-sided rejection with the box's tightness.

The simplex is deliberately scoped here as a standalone primitive, usable with any spatial structure. Two companion ideas it composes with — a pointer-free rotating-axis sort that encodes a hierarchy in array order, and a SIMD-batch median layout that fuses lane-width granularity into that sort — are deferred to separate treatments, where they can be developed and evaluated on their own terms.

References

[1] A. S. Glassner, "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications*, 1984.

[2] J. T. Klosowski et al., "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," *IEEE Transactions on Visualization and Computer Graphics*, 1998.

[3] G. Zachmann, "Rapid Collision Detection by Dynamically Aligned DOP-Trees," in *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS)*, 1998, pp. 90–97.

[4] J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin, "An Optimal Algorithm for Finding Minimal Enclosing Triangles," *Journal of Algorithms*, vol. 7, no. 2, pp. 258–269, 1986.

[5] Y. Zhou and S. Suri, "Algorithms for a Minimum Volume Enclosing Simplex in Three Dimensions," *SIAM Journal on Computing*, vol. 31, no. 5, pp. 1339–1357, 2002.

[6] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[7] P. M. Hubbard, "Approximating Polyhedra with Spheres," *ACM Transactions on Graphics*, 1996.

[8] K. Perlin, "Improving Noise," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 21, no. 3, pp. 681–682, 2002.

[9] H. Weghorst, G. Hooper, and D. P. Greenberg, "Improved Computational Methods for Ray Tracing," *ACM Transactions on Graphics*, 1984.

[10] C. Ericson, *Real-Time Collision Detection*, Morgan Kaufmann, 2004.

[11] P. Jiménez, F. Thomas, and C. Torras, "3D Collision Detection: A Survey," *Computers & Graphics*, vol. 25, no. 2, pp. 269–285, 2001.

[12] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire, *Real-Time Rendering*, 4th ed., A K Peters/CRC Press, 2018.

[13] J. Goldsmith and J. Salmon, "Automatic Creation of Object Hierarchies for Ray Tracing," *IEEE Computer Graphics and Applications*, 1987.

[14] J. D. MacDonald and K. S. Booth, "Heuristics for Ray Tracing Using Space Subdivision," *The Visual Computer*, 1990.

[15] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *SIGGRAPH*, 1996.

[16] Intel Corp., "Intel 64 and IA-32 Architectures Optimization Reference Manual," 2023.

[17] T. Aila and S. Laine, "Understanding the Efficiency of Ray Traversal on GPUs," *High Performance Graphics*, 2009.