Enhanced K-Directional Convex-Hull Discovery with Adaptive Point-Pair Orientations, Envelope Search, and Certified Completion

Technical White Paper — Version 1_0_03

TargetDeLab Group

9 October 2026

Abstract

This work studies vertex discovery for the convex hull of a finite point set in $\mathbb{R}^d$, emphasizing regimes in which enumerating facets is expensive. An enhanced K-directional discovery procedure combines a banded linear transformation, exact upper/lower envelopes over a scalar parameter, adaptive point-pair Householder orientations, global extremum detection, a sufficient cross-polytope interior certificate, and linear-programming (LP) completion. We formulate the method mathematically, establish sufficient conditions for candidate extremality and hull-spanning completion in exact arithmetic, and carefully distinguish a hull-spanning candidate representation from a minimal vertex enumeration. We report exploratory measurements against a specific simplex-initialized LP-only implementation and discuss the limits of those comparisons. The method is not proved universally faster than LP or Qhull, and its numerical implementation requires independent minimality verification for general inputs.

Contents

Notation 3

Research objective and intuitive overview 4

1 The convex-hull problem 4

2 Established approaches and software 5

2.1 Classical methods 5

2.2 Toolboxes and comparison caveat 5

3 Mathematical construction of the enhanced method 5

3.1 Normalization and K-direction family 5

3.2 Adaptive point-pair Householder orientation 6

3.3 Global projections and envelope search 7

4 Completion and correctness 7

4.1 LP separation test 7

4.2 Sufficient interior certificate for antipodal points 8

5 Algorithm summary and implementation complexity 8

6 Experimental methodology and observed results 9

6.1 Scope and comparability 9

6.2 Dataset construction and reproducibility 9

6.3 Selected large-scale measurements 10

6.4 Interpretation of high-dimensional Gaussian data 12

7 Discussion: where superiority is and is not established 12

7.1 Algorithmic comparison and interpretation 12

7.2 Numerical limitations and failure modes 12

8 Conclusions 13

9 FUTURE WORK 13

9.1 Algorithmic enhancements and computational scalability 13

9.2 Mathematical and numerical validation 14

9.3 Potential engineering and scientific applications 14

9.4 Application-oriented benchmark program 15

Notation

Symbol	Meaning
N, d	Number of input points and ambient dimension
X	Input point set; normalized row-wise matrix where specified
S_0, S	Discovered candidates and candidates after completion
U	Input points not yet selected by directional discovery
B	SVD-evaluated $X(X^\top X)^{-1}$ when full column rank holds
$A, K(\mu)$	Banded matrix and $I_N - \mu A$
H_{ij}	Pair-associated Householder transform
t^*, ε	Optimal LP separation margin and numerical tolerance

Research objective and intuitive overview

The input is a collection of points in a potentially high-dimensional space. The goal is to return a smaller subset of those *same input points* whose convex hull equals the hull of the complete dataset. This is a vertex-discovery and hull-coverage problem, not necessarily a facet-enumeration problem. In practical terms, the algorithm asks: *which observed points are sufficient to reproduce the entire convex region?*

The method first probes the data using structured families of linear directions. Points that attain extreme projections become candidates. Additional orientations adapt the search toward points that have not yet been discovered. Finally, an LP separation test examines whether the remaining points lie within the candidate hull; points that can be separated trigger further additions. In exact arithmetic, exhaustive successful completion proves hull equivalence. It does *not* by itself remove all redundant candidates.

A small geometric example (illustrative, not a benchmark). Consider twelve points in the plane: $(0,0), (4,0), (4,3), (0,3)$ and the eight interior points $(1,1), (2,1), (3,1), (1,2), (2,2), ($ The hull is the rectangle determined by the first four points; the remaining eight do not change it. An ideal minimal vertex enumeration returns precisely those four corners. A valid hull-spanning output may also contain some interior points; the completion criterion only guarantees that the output spans the rectangle. This toy example explains the intended output but does not assert a measured result of the released code.

Why directional extrema? A vertex of a finite point set can be singled out by a suitable linear functional. The K-direction family and its adaptive Householder orientations search many such functionals, while affine upper/lower envelopes locate extrema over a continuous parameter interval without sampling every parameter value. The LP stage provides a separate coverage check. The discovery stage is heuristic; the exact-arithmetic completion argument supplies the hull-spanning guarantee under its explicit hypotheses.

Scope of the claim. The contribution is a hybrid discovery–completion procedure. Its output is a hull-spanning candidate set, possibly containing redundant nonvertices. Numerical timings are evidence about the tested implementations and datasets, not a general complexity or minimal-vertex theorem.

1 The convex-hull problem

Let $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ and let $\mathbf{X} \in \mathbb{R}^{N \times d}$ contain $x_i^\top$ in row i. The convex hull is

$$\operatorname{conv}(X) = \left\{ \sum_{i=1}^{N} \lambda_i x_i : \lambda_i \geq 0,\ \sum_i \lambda_i = 1 \right\}. \tag{1}$$

A point x_j is a *vertex* if $x_j \notin \mathrm{conv}(X \setminus \{x_j\})$. For a finite point set, a vertex is exposed: there exists $w \in \mathbb{R}^d$ with $w^\top x_j > w^\top x_i$ for every $i \neq j$. Repeated or nearly coincident points and rank-deficient configurations demand special numerical treatment.

Two distinct outputs must not be conflated. A *vertex enumeration* gives exactly the extreme points. A *hull-spanning candidate set* $S \subseteq X$ satisfies $\mathrm{conv}(S) = \mathrm{conv}(X)$ but can include redundant nonvertices. Facet enumeration additionally returns the supporting faces and incidence information; its computational task is not equivalent to vertex-only extraction.

2 Established approaches and software

2.1 Classical methods

Planar Graham scan sorts points by polar order and constructs the boundary using orientation tests; divide-and-conquer and gift-wrapping methods provide other low-dimensional approaches [2]. Quickhull partitions points by visible faces, extending a hull incrementally. Its higher-dimensional implementation combines Quickhull with beneath–beyond construction [1]. Randomized incremental and output-sensitive methods have distinct complexity guarantees depending on dimension and output size [3]. The double-description approach converts between vertex and inequality representations [4]. Linear programming can test whether a point is in the convex hull of others, or construct separating hyperplanes, without enumerating all facets [5].

2.2 Toolboxes and comparison caveat

Qhull is a widely used implementation [6]. SciPy's `scipy.spatial.ConvexHull` and MATLAB's `convhulln` use Qhull, so treating them as independent algorithm families would overstate the breadth of a comparison [7, 8]. MATLAB `linprog` and SciPy `scipy.optimize.linprog` implement numerical optimization approaches; LP-only is a practical computational baseline, not merely a theoretical construction. A fair comparison must specify whether each method returns vertices, facets, or both, and how tolerances, initialization, compilation, and timing are handled.

3 Mathematical construction of the enhanced method

3.1 Normalization and K-direction family

For numerical scaling, define $\bar{x} = N^{-1} \sum_i x_i$ and $s = \max_{i,k} |x_{ik} - \bar{x}_k| > 0$. Use $\widetilde{x}_i = (x_i - \bar{x})/s$ and write X for the normalized matrix in the following derivation. Assume

rank$(X) = d$; the present implementation does not provide a general lower-dimensional affine-hull reduction.

Define

$$B = X(X^\top X)^{-1} \in \mathbb{R}^{N\times d}, \qquad B^\top X = I_d. \tag{2}$$

The implementation evaluates B by an SVD rather than explicitly forming $(X^\top X)^{-1}$. Let

$$A_{ij} = \begin{cases} 1, & |i-j| < d, \\ 0, & \text{otherwise}, \end{cases} \qquad K(\mu) = I_N - \mu A, \qquad D(\mu) = K(\mu)B. \tag{3}$$

The scalar interval inherited from the original degree-of-freedom schedule is

$$\mu \in [\mu_-, \mu_+], \quad \mu_- = \frac{1-d}{1+d}, \quad \mu_+ = \frac{N-1}{N+1}. \tag{4}$$

The original discrete schedule used Nd values. The enhanced procedure instead searches the continuous interval through exact envelopes of affine score functions. Thus it retains the K family and parameter interval, but not the original iteration schedule.

3.2 Adaptive point-pair Householder orientation

Let (i, j) be an eligible pair of input points. Define normalized vectors

$$a = \frac{x_i - x_j}{\|x_i - x_j\|_2}, \qquad b = \frac{B_i - B_j}{\|B_i - B_j\|_2}. \tag{5}$$

If either denominator is zero, the pair is skipped. Otherwise let $v = (b-a)/\|b-a\|_2$ and

$$H_{ij} = I_d - 2vv^\top. \tag{6}$$

If $a = b$ numerically, use $H_{ij} = I_d$. The directional matrix for the pair is

$$C_{ij}(\mu) = (B - \mu AB)H_{ij}. \tag{7}$$

The current default evaluates at most two pairs. The first pair combines a point of maximal squared norm with its farthest partner. Let S_0 denote the indices discovered by the directional search so far and define the undiscovered working set $U = \{x_k : k \notin S_0\}$. A *U–U pair* is a pair (x_i, x_j) with both points in U. Subsequent pairs are selected by maximal Euclidean separation within U (the *U–U* policy). The set U is not a set of certified nonvertices: it merely contains points not yet selected by discovery. This policy is a heuristic; it does not guarantee that a prescribed number of pairs discovers all vertices.

3.3 Global projections and envelope search

For each row r of $C_{ij}(\mu)$ and every input point x_k, define

$$s_{rk}(\mu) = [C_{ij}(\mu)]_r \, x_k = \alpha_{rk} - \mu\beta_{rk}, \tag{8}$$

where $\alpha_{rk} = (BH_{ij})_r x_k$ and $\beta_{rk} = (ABH_{ij})_r x_k$. For each r, construct the upper and lower envelopes

$$u_r(\mu) = \max_{1\le k\le N} s_{rk}(\mu), \qquad \ell_r(\mu) = \min_{1\le k\le N} s_{rk}(\mu). \tag{9}$$

The union of all maximizing and minimizing indices attained over $[\mu_-, \mu_+]$ is marked as the discovery set S_0. Importantly, *all* N input points participate in each extremum comparison, including points used in earlier orientations. The implementation sorts affine functions by slope and maintains an envelope stack; compiled Numba routines accelerate the Python version. Ties, collinearity, and floating-point slope comparisons need particular care.

Proposition 3.1 (Strict extremum implies vertex)**.** *If x_j uniquely maximizes $w^\top x_k$ over all input points for some w, then x_j is a vertex of* $\mathrm{conv}(X)$.

Proof. If $x_j = \sum_{k\neq j} \lambda_k x_k$ with $\lambda_k \ge 0$ and $\sum \lambda_k = 1$, then $w^\top x_j = \sum_{k\neq j} \lambda_k w^\top x_k < \sum_{k\neq j} \lambda_k w^\top x_j = w^\top x_j$, a contradiction. □

Remark 3.2. *A nonunique MIN–MAX winner need not be a vertex: a point in the relative interior of an exposed face can tie for the same projection. The implementation does not automatically certify strictness for every discovered candidate.*

4 Completion and correctness

4.1 LP separation test

For a current candidate set S and an unexamined point x_j, consider

$$\max_{w,t} \quad t \tag{10}$$

$$\text{subject to} \quad w^\top(x_j - x_k) \ge t \quad (x_k \in S), \tag{11}$$

$$-1 \le w_m \le 1 \quad (m = 1, \ldots, d). \tag{12}$$

Denote the optimum by $t^*(j, S)$. Since $w = 0, t = 0$ is feasible, $t^* \ge 0$. A positive optimum gives a separating direction, and maximizing $w^\top x_k$ over the *entire* dataset identifies an exposed candidate to add. Iterate until every unselected point has a nonpositive optimum, subject to numerical tolerance.

Theorem 4.1 (Exact-arithmetic hull spanning)**.** *Suppose LPs are solved exactly, every point with positive separation is processed, the algorithm adds a maximizer over the complete dataset whenever positive separation occurs, and it terminates only when* $t^*(j, S) = 0$ *for every* $x_j \notin S$*. Then* $\mathrm{conv}(S) = \mathrm{conv}(X)$*.*

Proof. The inclusion $\mathrm{conv}(S) \subseteq \mathrm{conv}(X)$ is immediate. If some $x_j \notin \mathrm{conv}(S)$, strict separation of a point and a compact convex polytope gives a vector $w \neq 0$ and $\delta > 0$ with $w^\top(x_j - x_k) \geq \delta$ for all $x_k \in S$. Rescale w so $\|w\|_\infty \leq 1$; the LP then has positive feasible margin, contradicting termination. Hence all $x_j \in \mathrm{conv}(S)$, giving the reverse inclusion. □

Remark 4.2 (Numerical qualification)**.** *In floating point, the condition* $t^* \leq \varepsilon$ *is an approximate separation certificate, not the exact theorem's hypothesis. Solver feasibility tolerances, conditioning, and nearly degenerate configurations can affect classification. The released implementation also retains initial discovered candidates, so the theorem does not imply minimal vertex enumeration.*

4.2 Sufficient interior certificate for antipodal points

Suppose the current selected set contains $c \pm v_1, \ldots, c \pm v_d$ with linearly independent v_k, and write $V = [v_1 \ \cdots \ v_d]$. The cross-polytope generated by these points is

$$\mathcal{C} = \{c + Va : \|a\|_1 \leq 1\}. \tag{13}$$

Proposition 4.3 (Certified containment)**.** *If* $x = c + Va$ *and* $\|a\|_1 \leq 1$*, then* $x \in \mathrm{conv}\{c \pm v_1, \ldots, c \pm v_d\} \subseteq \mathrm{conv}(S)$*.*

Proof. Write $a_k = a_k^+ - a_k^-$ with $a_k^\pm \geq 0$. The total mass $\sum_k (a_k^+ + a_k^-) = \|a\|_1 \leq 1$. Assign masses a_k^+ and a_k^- to $c + v_k$ and $c - v_k$, respectively, and distribute the remaining mass equally to any antipodal pair. The resulting convex combination equals $c + Va$. □

The numerical filter searches for approximately antipodal selected pairs and a full-rank basis, solves for coefficients, and uses a conservative threshold below one. Its validity depends on accurate antipodal identification and linear solves. It is a *sufficient*, not necessary, certificate and is not a general-purpose interior oracle.

5 Algorithm summary and implementation complexity

Input: $N \times d$ real point matrix; orientation budget q (default 2); normalized LP tolerance ε.

Step 1. Validate and normalize input; compute SVD-based B and banded product AB without materializing A or K.
Step 2. Select an initial point pair, construct H_{ij}, and compute the affine score coefficients.
Step 3. Evaluate upper/lower envelopes for every direction row; mark MIN–MAX winners over all N points.
Step 4. Repeat with U–U pair selection up to budget q.
Step 5. Apply the sufficient cross-polytope containment certificate where possible.
Step 6. Use LP separation for remaining unexamined points; add globally exposed points until no separation remains.
Output: A hull-spanning candidate set, diagnostic timings, LP counts, and pair indices. No general minimality guarantee is asserted for the returned index set.

The matrix B requires an SVD with cost dependent on N, d; a conventional thin SVD costs $O(Nd^2)$ for $N \geq d$. Forming the pairwise distance matrix costs $O(N^2 d)$ time and $O(N^2)$ storage. The precomputed products $BX^\top$ and $(AB)X^\top$ are $N \times N$ arrays and require $O(N^2 d)$ arithmetic by dense multiplication and $O(N^2)$ storage. Each of q orientations requires updates to these score matrices and envelope sorting, on the order of $O(qN^2 \log N)$ in a straightforward implementation. LP-completion cost depends strongly on the number of calls and solver behavior; no universal total-runtime superiority theorem follows from this decomposition. The dense $N \times N$ intermediate matrices are a significant limitation for much larger N.

6 Experimental methodology and observed results

6.1 Scope and comparability

The recorded benchmarks compare the enhanced algorithm with a particular *simplex-initialized LP-only completion implementation*, not all LP solvers or all conceivable LP-only designs. Both use a related LP completion routine, but the enhanced method begins with K-discovered candidates and may skip LP calls via a containment certificate. Timings below are exploratory single runs unless otherwise stated; they exclude the cost of initial JIT compilation and are not robust confidence intervals. The selected-set agreement flag compares two implementations, not an independent proof that each output is minimal. Hardware and complete solver-version metadata are not consistently retained in the available benchmark CSVs, limiting reproducibility claims.

6.2 Dataset construction and reproducibility

The benchmark driver used NumPy's `default_rng(seed)` with fixed integer seeds, one timed run per completed case, and three dataset classes. For *Gaussian* cases, the input

was an $N \times d$ array of independent standard-normal draws. For *clustered* cases, four independent standard-normal center vectors were generated; each input point selected one center uniformly among the four and added independent normal noise multiplied by 0.12. For the recorded *cross-polytope* cases, the generator stacked I_d, $-I_d$, and $(N - 2d)$ vectors whose coordinates were independently uniform on $[-0.025, 0.025]$. This last generator does *not* guarantee that every added point lies inside the unit ℓ_1 ball; consequently the number of genuine vertices need not equal $2d$. The cross-polytope benchmark must not be confused with the separate *guaranteed-interior* construction $z \mapsto 0.7z/\max(\|z\|_1, 10^{-15})$.

The displayed cases used seed 23 for Gaussian and clustered clouds and seed 11 for the cross-polytope case. The driver configured OpenBLAS, OpenMP, MKL, and Numba to use one thread each, warmed the compiled implementation, and measured the enhanced and QR-simplex-initialized LP-only methods. Each listed observation is a single run; no confidence intervals are available. The complete software version, processor, and solver configuration should be recorded in future repeated studies, along with code revision, input checksums, LP tolerances, timing scope, number of repetitions, and independent vertex checks. These are recommendations, not claims that such metadata were collected for the present table.

6.3 Selected large-scale measurements

Table 2: Measured benchmark comparisons. Lower runtime is better. The 500/70D cross-polytope benchmark returned 149 candidates; the random additions are not guaranteed interior.

N	d	Geometry	Enhanced (s)	LP-only (s)	Ratio
500	50	Clustered	0.149	2.829	19.0
500	70	Gaussian	0.122	3.986	32.7
1000	70	Gaussian	0.543	16.468	30.3
500	100	Gaussian	0.126	5.857	46.6
500	70	Cross-polytope	0.202	14.391	71.2

Reading Figure 2. Each bar compares two executions on the same dataset: its length is $T_{\mathrm{LP-only}}/T_{\mathrm{enhanced}}$, not an absolute execution time. The observed factors range from about 19 to 71 in these five recorded runs. The cross-polytope case has the largest ratio, but its special geometry and containment behavior differ from generic clouds. The figure therefore illustrates *case-specific measured advantages* over the chosen LP-only baseline; it cannot establish universal superiority, scaling laws, or statistical confidence from single-run data.

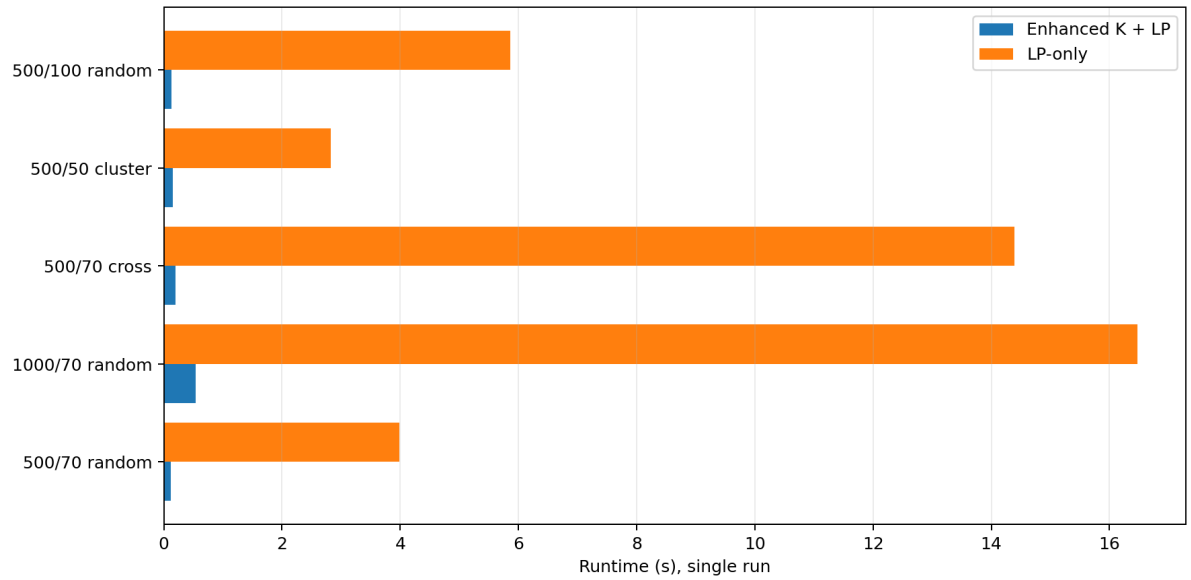


Figure 1: Measured execution times; geometry and point count are shown on the vertical axis.

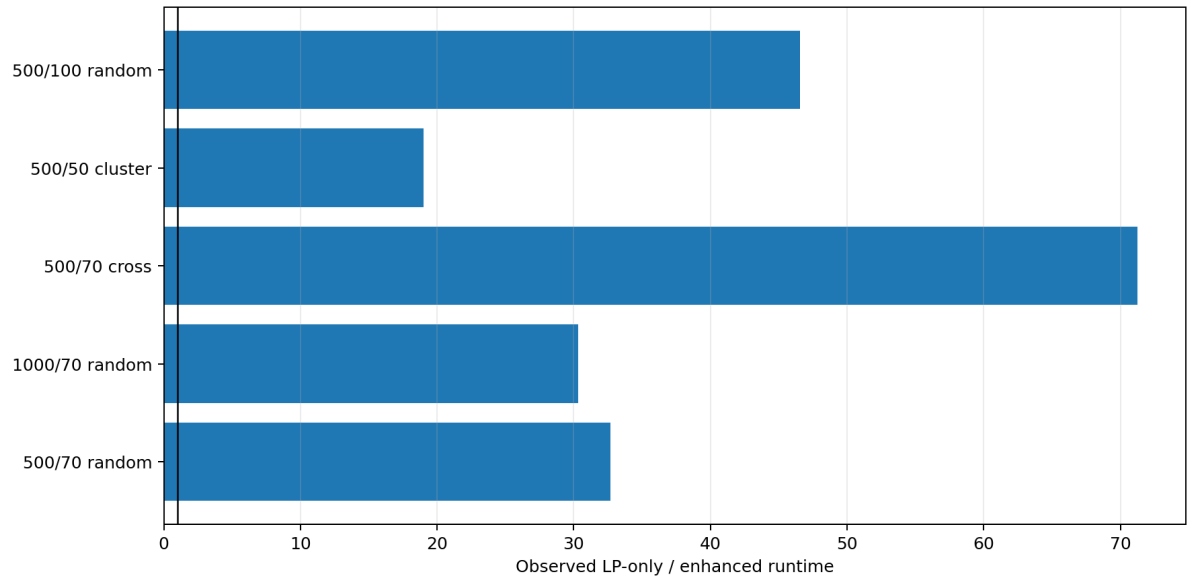


Figure 2: Figure 2: measured ratio of LP-only runtime to enhanced runtime for the five benchmark cases. The horizontal axis is the runtime ratio (speedup factor), and each category on the vertical axis specifies a dataset geometry, point count, and dimension. Values above one indicate lower measured runtime for the enhanced implementation.

6.4 Interpretation of high-dimensional Gaussian data

In the listed Gaussian-cloud cases, K discovery selected every input point and completion used zero LP calls. This is a strong timing advantage against the tested LP-only baseline, but a zero-call result is not a certificate of minimality. In sufficiently high dimension with finite samples, many randomly generated points may indeed be vertices, yet this must be checked for each dataset rather than assumed. In cross-polytope cases, the specialized containment certificate can avoid numerous LP checks when its structural hypotheses are satisfied.

7 Discussion: where superiority is and is not established

7.1 Algorithmic comparison and interpretation

Table 3: Comparison of the tested algorithmic strategies; these are properties of the stated implementations, not universal guarantees.

Criterion	Enhanced hybrid	Tested LP-only baseline
Initial selection	Directional extrema	QR-simplex initialization
Coverage mechanism	LP completion when needed	LP-based tests
Output minimality	Not generally certified	Not independently certified
Runtime advantage	Dataset-dependent	Dataset-dependent

7.2 Numerical limitations and failure modes

Repeated or nearly coincident points can produce projection ties; a tied extremum need not be a vertex. Affine-rank deficiency violates the full-column-rank derivation unless the data are first reduced to their affine span. Near-singular linear algebra, approximately antipodal pair detection, and finite LP feasibility tolerances can affect classifications. Dense $N \times N$ score intermediates require quadratic storage. A comprehensive validation study should vary random seeds, dimensions, point configurations, and solver tolerances, and independently test every reported candidate for redundancy. These are outstanding validation tasks, not experiments claimed to have been completed here.

The data support the following *bounded empirical claim*: on the recorded large-scale benchmark cases, the enhanced implementation executed faster than the particular LP-only baseline, sometimes by more than one order of magnitude. This advantage is largely explained by K discovering many candidates before LP completion, and by the cross-polytope interior certificate reducing the number of LP solves.

The data do *not* establish universal algorithmic superiority, superior worst-case complexity, or superiority to optimized alternative LP formulations. Qhull returns facet information as well as vertices; comparisons to Qhull are therefore workload-dependent. The current code's returned candidate set is not guaranteed minimal without a separate redundancy check. Near-degenerate and affine-rank-deficient inputs, duplicate points, LP tolerances, and JIT warm-up require further controlled testing.

8 Conclusions

We developed a hybrid convex-hull discovery and completion method combining K-directions, continuous parameter envelopes, adaptive U–U point pairs, and LP-based hull completion. In exact arithmetic, the separation-based termination criterion establishes hull equivalence. The antipodal containment test gives an additional sufficient certificate in suitable geometries. Benchmark results are encouraging against the tested LP-only implementation, particularly when the discovery phase makes completion unnecessary.

9 FUTURE WORK

The following directions are proposed research questions rather than capabilities or performance advantages established by the current experiments. They should be evaluated against appropriate optimized baselines with documented hardware, tolerances, and datasets.

9.1 Algorithmic enhancements and computational scalability

An immediate objective is to reduce the quadratic storage associated with dense $N \times N$ intermediates, for example through blocked or matrix-free operations, without changing the mathematical discovery criterion. Adaptive selection of U–U point pairs, alternative orientation policies, and more selective parameter-envelope searches may improve discovery efficiency. The associated overhead must be measured: additional orientations are useful only when they reduce the total cost of discovery and completion. Other investigations include warm-started or batched LP completion, parallel CPU or GPU implementations, and the feasibility of FPGA acceleration for repetitive projection operations.

A separate objective is to certify *minimality* of the returned point set, rather than only hull equivalence. This requires an independent redundancy test or an equivalent proof for each retained candidate. Streaming and incremental variants could investigate whether previously discovered candidates and completion certificates can be reused when new input points arrive; no such update guarantee is presently claimed.

9.2 Mathematical and numerical validation

Future work should characterize the families of point sets and directional searches for which all extreme points are discovered before LP completion. Numerical investigations should address near-duplicate points, projection ties, affine-rank deficiency, ill-conditioned transformations, and tolerance-sensitive LP feasibility decisions. Affine-hull reduction and higher-precision checks may provide useful diagnostic controls. Repeated multi-seed studies should report runtime distributions, peak memory, LP call counts, candidate redundancy, and independently verified hull coverage. Comparisons should include optimized Qhull and well-engineered LP-based alternatives, while recognizing that algorithms producing facets and those producing only a spanning vertex subset solve different output tasks.

9.3 Potential engineering and scientific applications

Nonlinear control and reachability. For a collection of simulated states $X(t) = \{x^{(1)}(t), \ldots, x^{(N)}(t)\}$, a hull-spanning subset $S(t)$ could provide a compact representation satisfying $\mathrm{conv}(S(t)) = \mathrm{conv}(X(t))$. This identity concerns the *sampled* states only; it is not a certified outer bound for all reachable states of a continuous dynamical system. Possible uses include scenario-based analysis and geometric preprocessing in robust control.

Robotics and geometric processing. Candidate discovery may be studied as a preprocessing stage for convex approximations of obstacle point clouds, spatial extents, and collision-checking geometry. Application-specific tests must establish whether any reduction in point count offsets discovery overhead, particularly in conventional two- and three-dimensional cases where existing methods are highly optimized.

Machine learning and data analysis. Potential investigations include geometric reduction of large feature datasets, boundary-point identification, and preprocessing for methods that depend explicitly on convex combinations. High-dimensional feature geometry, noise sensitivity, and the proportion of true extreme points will determine practical usefulness.

Optimization and signal processing. Compact hull-spanning representations may assist polyhedral preprocessing and geometric analysis of multidimensional signal features. Any speedup of downstream optimization or detection tasks must be measured end to end, rather than inferred from the hull-reduction step alone.

9.4 Application-oriented benchmark program

A structured program should compare Gaussian and uniform point clouds, clustered data, deliberately constructed polytopes with known vertices, and nearly degenerate configurations over increasing N and d. Each application study should report both the quality of the returned representation and the full computational cost, including preprocessing, directional discovery, LP completion, and any minimality check. The objective is to identify reproducible regimes of advantage and regimes where established methods remain preferable.

References

[1] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," *ACM Transactions on Mathematical Software*, 22(4), 469–483, 1996. DOI: 10.1145/235815.235821.

[2] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed., Springer, 2008.

[4] K. Fukuda and A. Prodon, "Double Description Method Revisited," in *Combinatorics and Computer Science*, LNCS 1120, 1996.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[6] Qhull Project, *Qhull documentation*, https://www.qhull.org/.

[7] SciPy Developers, *scipy.spatial.ConvexHull documentation*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html.

[8] MathWorks, *convhulln documentation*, https://www.mathworks.com/help/matlab/ref/convhulln.html.