Open Access

Eigenvector Sign Ambiguity in ITK's Oriented Bounding Box: A Latent Representation Defect Exposed by the VNL-to-Eigen Migration

Why the ShapeLabelObject oriented-bounding-box test flakiness is not a tolerance problem, and what a robust fix looks like

Hans J. Johnson[1]

[1]Department of Electrical and Computer Engineering, University of Iowa

Abstract

ITK's `itk::ShapeLabelObject` reports an oriented bounding box (OBB) whose "origin" corner is derived from the eigenvectors of the inertia tensor. The sign of a symmetric-matrix eigenvector is mathematically arbitrary, so the reported origin corner is not a portable quantity: two valid eigendecompositions that differ only by a column-sign flip produce the *identical* box but report *different* corners as the origin. The cross-platform flakiness of the `ShapeLabelMapFixture *_Direction` tests (issue #6518, PR #6521) is a manifestation of this representation defect — latent in the OBB since 2017 and re-exposed, not introduced, by the VNL/EISPACK-to-Eigen eigensolver migration (PR #6475), which changed which arbitrary corner is selected. The box geometry (size, vertex set, center) is invariant; only the origin label is not. We propose a fix that makes the OBB geometry reproducible: in the tests, assert the invariant vertex set and the frame-invariant OBB *center*; in the API, expose the OBB center and document the origin corner as non-portable. All source, data, parameters, and figure-generation scripts are included so the results regenerate from the bundle.

Keywords reproducible research, ITK, oriented bounding box, principal component analysis, eigenvector sign ambiguity, floating-point portability

0.a. TL;DR

- ITK's `ShapeLabelObject` reports an OBB **origin corner** computed from the inertia tensor's eigenvectors. An eigenvector's sign is mathematically arbitrary, so that corner is **not portable** across solvers, compilers, and platforms — the box is identical, only the corner labelled "origin" differs.
- The cross-platform `*_Direction` test flakiness is this defect. It is **latent in the OBB since 2017** and was **re-exposed, not introduced,** by the 2026 VNL-to-Eigen migration (PR #6475), which changed *which* arbitrary corner is reported. It is **not** a tolerance problem.
- **Fix:** in the tests, assert the invariant vertex set and the frame-invariant OBB **center**; in the API, expose the OBB center and document the origin as non-

Published Jun 27, 2026

portable. Loosening tolerances, re-picking a "canonical" corner, or switching to a minimum-volume box do not work and are not recommended.

0.b. Introduction

A cross-platform test failure in ITK's continuous integration — `ShapeLabelMapFixture.3D_T3x2x1_Direction` and `ShapeLabelMapFixture.3D_T2x2x2_Spacing_Direction`, the highest-volume flaky-test signal in the 2026 audit (issue #6518, 187 failures over 30 days) — presents as a tolerance problem: the failing assertions compare a computed oriented-bounding-box (OBB) *origin* against a baseline at `1e-4`.

It is not a tolerance problem. The OBB origin corner is derived from the mathematically sign-arbitrary eigenvectors of the inertia tensor, so it is not a portable quantity at all: the box geometry is invariant across eigensolvers and platforms, but which corner is reported as "origin" is not. The non-uniqueness is intrinsic to the representation and has been present since the OBB was added in 2017; the 2026 VNL-to-Eigen eigensolver migration (PR #6475) re-exposed it by changing which arbitrary corner is selected. This paper makes the mechanism precise, characterizes which objects trigger it, and proposes a fix that makes the OBB geometry reproducible.

0.c. The defect: a corner derived from sign-arbitrary eigenvectors

ITK's `ShapeLabelObject` OBB is the classic principal-axis box: the eigenvectors of the pixels' second-moment (inertia) tensor are the principal axes, and the box is the pixels' extent in that rotated frame (Gottschalk et al., 1996; Lehmann, 2007). For a symmetric matrix, only the *axis* each eigenvector spans is determined; its sign is mathematically free, so replacing a column v by $-v$ is an equally valid eigendecomposition — the well-known PCA/SVD sign ambiguity (Bro et al., 2008). (A separate, deeper non-uniqueness arises when two eigenvalues are near-equal; it is developed in Appendix B.)

In `itkShapeLabelMapFilter.hxx` the principal axes are these eigenvectors (`SymmetricEigenDecomposition<double>`, line 310; `principalAxes = eigen.V.transpose()`, line 317), with a determinant reflection that flips only the *last* axis to guarantee a proper rotation (lines 330–332). `ComputeOrientedBoundingBox` then projects the pixels into that frame and back-transforms the minimum corner to produce the reported origin (line 878). The origin is therefore a direct function of the eigenvector signs; the reflection fixes handedness but not each axis's absolute sign.

Figure 1 shows the consequence for the `T2x2x2_Spacing_Direction` case: the origin reported by the Eigen path and the origin reported by the prior VNL/EISPACK path are *diagonally opposite corners of the identical box*. The box — its size, orientation, and full set of eight vertices — is the same; only the corner selected as "origin" differs. The two origins differ by the full diagonal of the box, an $O(\text{object size})$ relabeling, not an $O(\text{tolerance})$ drift.

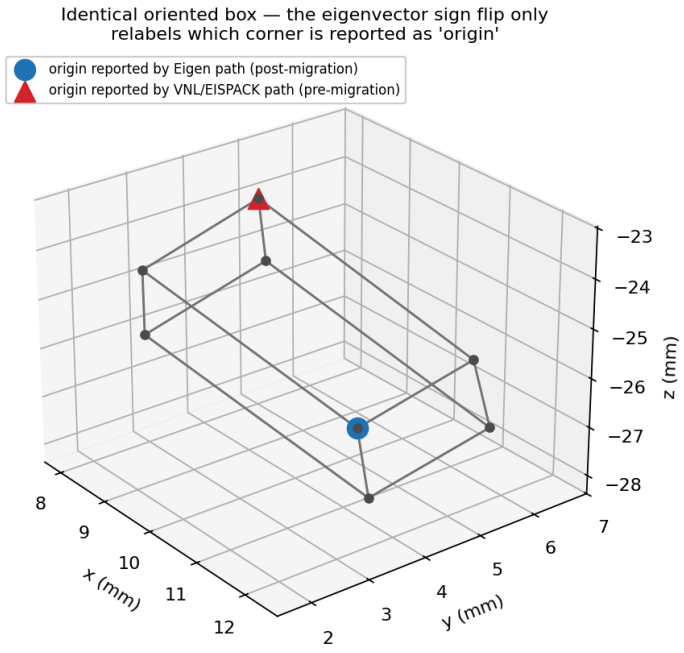

Figure 1: The reported OBB origin is not a portable quantity. The VNL/EISPACK and Eigen eigensolvers produce the *same* oriented box for the `T2x2x2_Spacing_Direction` test object, but a column-sign difference in the eigenvectors makes them report *different corners* as the origin. The geometry is invariant; the corner label is not. Rendered from `data/obb_vertices.csv` and `data/platform_origins.csv`.

0.d. A latent defect, not a regression

The 2026 migration (PR #6475) replaced EISPACK with `itk::SymmetricEigenDecomposition` and added a deterministic sign canonicalization (`itk::detail::CanonicalizeEigenvectorColumnSigns`, pinning each eigenvector column's largest-magnitude entry positive). This made the eigendecomposition bit-reproducible across the platforms ITK tests, but it *changed which corner* the OBB reports, because the canonical signs differ from whatever signs EISPACK emitted. The same change forced a one-time baseline regeneration in the sibling `LabelGeometryImageFilter` (commit `ad1af618f54`). Table 1 shows the analogous shift for the two ShapeLabelMap `*_Direction` cases: the origin moves to a different corner while the box **size** is unchanged.

Test case	VNL/EISPACK origin	Eigen origin	OBB size (unchanged)
T3x2x1_Dir	(3.22524, −3.19685, −14.83670)	(6.02222, −4.47691, −15.57049)	(1, 2, 3)
T2x2x2_Spc_Dir	(8.92548, 4.27240, −23.31018)	(9.75747, 5.36134, −28.03480)	(2.0, 2.2, 4.4)

Table 1: The VNL-to-Eigen migration shifted the reported OBB origin to a different corner of the same box; the box **size** (rotation-invariant) did not change. Pre-migration values are the 5-decimal EISPACK baselines from before PR #6475. Cases abbreviated: Dir = Direction, Spc = Spacing. From `data/platform_origins.csv`.

The migration replaced one deterministic-but-arbitrary corner choice with another; the defect is in neither eigensolver but in the *representation*, where a single corner derived from sign-free eigenvectors is not a portable invariant. The non-uniqueness is not new. The OBB-origin assertion was disabled in the test suite in 2017 (commit `0d9a5ed6`) precisely because, in its author's words, *"the solution for the OBB is not unique due to ambiguity in the sign of the eigen vectors for the axes,"* producing different results across architectures. That workaround remained in place for nine years; Appendix A gives the factual record. The present flakiness is the same defect surfacing through the two `*_Direction` cases, which assert the origin rather than an invariant.

One distinction matters for scoping. The 2026 flakiness has two separate causes: three non-`Direction` cases fail Windows-only under MSVC floating point — a precision effect on rotation-invariant scalars — while only the two `*_Direction` cases fail cross-platform through the principal-axes / OBB-origin path. Only the second is the sign-ambiguity defect addressed here; the first is a distinct numerical concern (Appendix B) and is not folded into this fix.

0.e. The box is invariant; only the corner label is not

The geometric content of the OBB is reproducible across platforms; only the chosen corner is not. Three lines of evidence establish this. First, across the migration the box is identical: each pre-migration origin coincides with a post-migration *vertex* to ∼1e-5 (the precision of the old 5-decimal baseline), so the migration relabeled the origin rather than moving the box. Second, three independent current toolchains — Apple clang/arm64, GCC 13/x86-64 at all SIMD levels, and MSVC 14.38/x64 — land on the *same* post-migration corner to ∼1e-13 (Table 3). Third, the OBB size is stable to ∼ 1e-12 on all three; because a rotation within a degenerate eigenspace would change the size, its stability rules out such a rotation for these distinct-eigenvalue cases, leaving a sign relabeling as the only remaining variation. Which objects are susceptible, and why, is developed in Appendix B.

0.f. Recommended fix

The geometric content of an OBB is fully captured by quantities that are invariant to the eigenvector signs: the box **size**, the **axes up to sign and permutation**, the **full vertex set**, and the box **center** (the transformed-back midpoint of the extents). The single non-portable scalar is the origin corner, and it is *redundant* given center, size, and axes. The fix follows directly.

Tests. Replace the exact `GetOrientedBoundingBoxOrigin()` assertion in the two `*_Direction` cases with a membership check against the invariant vertex set, `TestListHasPoint(GetOrientedBoundingBoxVertices(), expectedCorner)` — the pattern ITK already uses for the sibling `3D_T2x2x2_Spacing` case — and add an

assertion on the frame-invariant OBB **center**, keeping the existing size assertions. Size + vertex membership + center fully constrain the box while asserting only invariant quantities; this tightens, rather than loosens, the geometric checks.

API. Expose the OBB **center** as a first-class accessor (`GetOrientedBoundingBoxCenter()`), and **document `GetOrientedBoundingBoxOrigin()` as an arbitrary, non-portable corner**, directing users to the vertex set or the center for reproducible geometry. The center is computed as origin $+ \frac{1}{2} A^T s$ for box size s and principal-axis matrix A; it is invariant to the eigenvector signs and is *not* the inertia centroid (the box is not symmetric about the centroid). This change is additive and baseline-neutral. ITK's public contract is currently silent on the origin's canonicality (Insight Software Consortium, 2026), so documenting it fills a gap rather than breaking a promise.

Degeneracy. When two principal moments are near-equal, the box *orientation* is itself ill-conditioned (Appendix B); document this so callers know the axes — and therefore the origin corner — are not meaningful in that regime.

Three alternatives are **not** suitable and are noted to forestall them. Loosening the origin tolerance cannot work: the divergence is $O(\text{object size})$, not $O(\text{tolerance})$. Re-picking a "canonical" corner — e.g. the lexicographically smallest vertex — only relocates the same discontinuity to axis-aligned boxes (a more common configuration) and changes the public output for every object. Switching to a minimum-volume OBB changes the documented inertia-axis box for *all* objects (Dimitrov et al., 2009; O'Rourke, 1985) and still does not resolve the symmetric-object ambiguity. Table 2 summarizes.

Action	Recommendation	Reason
Vertex-set membership + center assertion in tests	**Adopt**	Asserts only invariant geometry; tightens the checks
Expose frame-invariant OBB center; document origin as non-portable	**Adopt**	Additive, baseline-neutral; makes geometry reproducible
Document degeneracy / ill-conditioning	Adopt	Matches the mathematics
Loosen the origin tolerance	Not recommended	Divergence is O(object size), not O(tolerance)
Lexicographic-min canonical corner	Not recommended	Relocates the discontinuity to axis-aligned boxes; changes API
Switch to minimum-volume OBB	Not recommended	Changes all objects; breaks the inertia-axis contract

Table 2: Recommended actions and alternatives not recommended.

0.g. Reproducibility

The bundle includes the standalone reproduction used for every measurement. `src/obb_repro.cxx` builds the three test objects through `itk::LabelImageToShapeLabelMapFilter` and dumps, at full `double` precision, the principal moments, principal axes, OBB size, origin, and all eight vertices; `src/sensitivity.cxx` performs the moment-tensor perturbation sweep. The committed CSVs in `data/` are the single source for every figure; `scripts/make_figures.py` reads only those CSVs and regenerates the figures. Measurement environments are stated next to every value (macOS arm64 / AppleClang; Linux x86-64 / GCC 13; Windows / MSVC 14.38; all Release or RelWithDebInfo, `ITK_USE_FLOAT_SPACE_PRECISION=OFF` unless noted). Building `src/` against ITK reproduces the `data/` values.

0.h. Conclusion

The ShapeLabelObject OBB test flakiness is a representation defect, not a precision problem: the reported origin corner is derived from sign-arbitrary eigenvectors and is not portable across solvers and platforms. The box geometry is invariant; only the corner label is not. The fix is to assert invariant geometry in the tests and to make the OBB geometry portable in the API by exposing the frame-invariant center and documenting the origin corner as non-canonical — rather than loosening tolerances, re-picking a corner, or redefining the box.

Appendices

Supporting material — not required reading

Review-history provenance · conditioning and cross-platform evidence

0.i. Appendix A — Provenance: the defect in the record since 2017

The OBB feature was added in May 2017 (commit `d0b6e92`). The sign non-uniqueness was identified the same year: commit `0d9a5ed6` (15 November 2017), *"BUG: Disable checking of OBB origin,"* disabled the `GetOrientedBoundingBoxOrigin()` assertion in the GoogleTest fixture with the explanation:

> The solution for the OBB is not unique due to ambiguity in the sign of the eigen vectors for the axes. This particular case does not produce the same results on 32-bit and 64-bit intel system due to floating point tolerances.

and left an in-source note whose descendant survives in the fixture today:

> [We omit] these because the sign of the eigen vectors is not unique, therefore the axes may not always point in the same direction and the origin may not be the same corner.

The workaround — disabling the assertion rather than making the output portable — remained in place for roughly nine years across dozens of style and dependency-resync commits. The `3D_T2x2x2_Spacing` case was later given a vertex-list membership check (2021) as a partial form of the same workaround; the `*_Direction` cases were not. The 2026 migration (PR #6475) was the first change to address the sign convention in production code, adding the deterministic canonicalization and regenerating the sign-dependent `LabelGeometryImageFilter` baselines. The defect is therefore long-standing and intrinsic to the representation, not a product of the migration.

0.j. Appendix B — Conditioning and cross-platform evidence

0.j.i. *Two sources of non-uniqueness:*

The eigenvectors carry two distinct non-uniquenesses. The first is **sign**: a labeling non-uniqueness that leaves the box invariant. The second is **degenerate-subspace rotation**: when two eigenvalues coincide, any rotation within their shared eigenspace is a valid eigenbasis, and the Davis–Kahan $\sin\theta$ theorem bounds eigenvector rotation by the perturbation magnitude divided by the spectral gap (Davis & Kahan, 1970; Golub & Van Loan, 2013), so as the gap shrinks the eigenvectors become arbitrarily ill-conditioned. Unlike a sign flip, a subspace rotation changes the box itself (different size and vertices). Because the rounding that drives these computations differs across platforms and compilers (Goldberg, 1991), a test of OBB output must assert quantities invariant under sign and robust under rotation — size, axis collinearity, the vertex set up to permutation — not one corner's coordinates.

0.j.ii. *Why these cases:*

Figure 2 shows the principal-moment spectra. The `T2x2x2_Spacing` family has two near-equal moments (0.25 vs 0.3025, a 4.3% spectral gap), the Davis–Kahan regime

where eigenvectors are most sensitive. The axis-aligned member has its eigenvectors on the coordinate axes, so the sign canonicalization is unambiguous and the case is stable; a non-identity image `Direction` rotates the inertia tensor, spreading the eigenvector components across axes and shrinking the margin by which the canonicalization picks a sign. The `T3x2x1_Direction` case instead has a well-separated spectrum but a near-zero smallest moment (a one-voxel-thick object). Both `*_Direction` cases sit where a small solver change can flip the canonical sign.

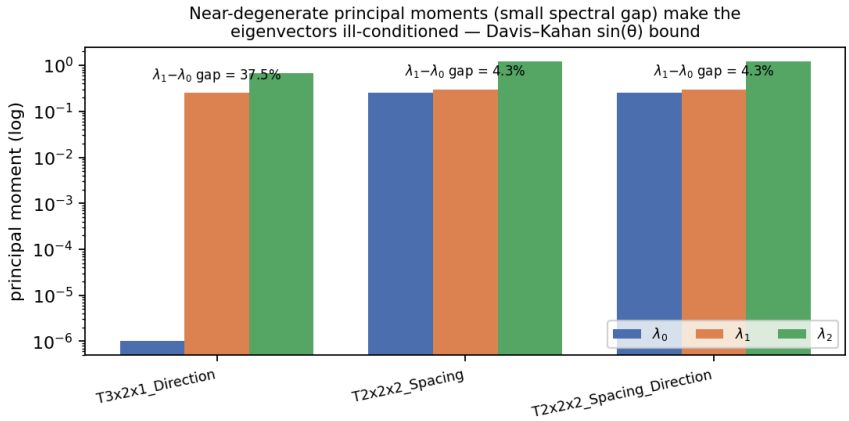

Figure 2: Principal-moment spectra of the three test objects (log scale). The near-degenerate 4.3% gap in the `T2x2x2_Spacing` family is the Davis–Kahan ill-conditioning regime for the eigenvectors. From `data/principal_moments.csv`.

0.j.iii. *Cross-platform agreement:*

Three current toolchains land on the same post-migration corner to ∼1e-13 (Table 3); the OBB size is stable to ∼1e-12 on all of them, which rules out a degenerate-subspace rotation (which would change the size) for these distinct-eigenvalue cases.

Test case	OBB origin (all three toolchains)	max pairwise Δ
T3x2x1_Dir	(6.0222153, −4.4769133, −15.5704904)	< 1e-12
T2x2x2_Spc_Dir	(9.7574735, 5.3613401, −28.0348041)	< 4e-13

Toolchains: Apple clang/arm64 (Release), GCC 13/x86-64 (Release, all SIMD levels), MSVC 14.38/x64 (RelWithDebInfo).

Table 3: Post-migration OBB origin is identical across three current toolchains to ∼1e-13. From `data/platform_origins.csv`.

0.j.iv. *Distinct, related numerical issues:*

Two distinct concerns share this filter but are not the sign-ambiguity defect and are out of scope for the fix. First, the Windows-only failures of the non-`Direction` cases are ∼5e-6 MSVC RelWithDebInfo FP-model differences on rotation-invariant scalars (`Perimeter`, spherical metrics, `Roundness`) — a genuine precision/baseline matter addressed separately by full-precision baselines, not an eigenvector-sign effect.

Second, the central moments are formed as $E[pp] - E[p]E[p]$ (`itkShapeLabelMapFilter.hxx:304`), the unstable variance form; for the `*_Direction` objects, which sit 15–28 units from the world origin while the moments are $O(0.5)$, this carries an amplification of $\sim (28/0.5)^2$ that can matter under `ITK_USE_FLOAT_SPACE_PRECISION=ON`. Neither is conflated with the OBB-origin fix.

Figure 3 shows that continuous perturbations of the moment tensor at rounding scale produce eigenvector drift far below the `1e-4` tolerance, so rounding within the eigensolve does not move a continuous result past the tolerance — consistent with the failure being a discrete sign relabeling rather than continuous drift.

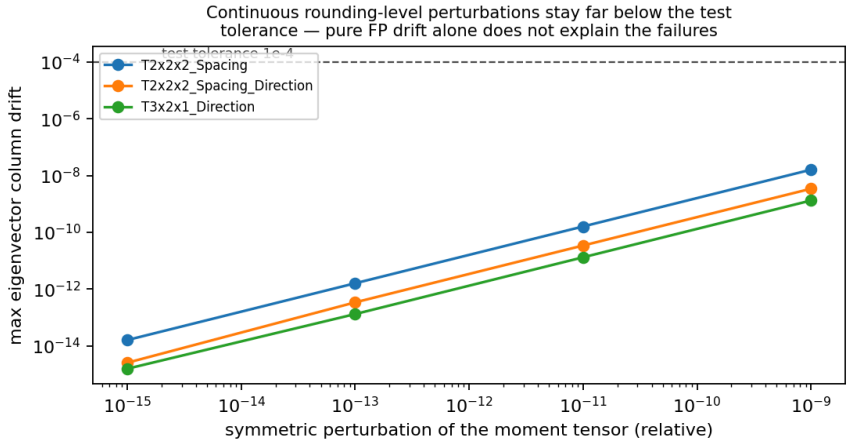

Figure 3: Eigenvector column drift versus a symmetric perturbation of the moment tensor. Even in the near-degenerate cases the drift stays far below the `1e-4` test tolerance. From `data/eigenvector_sensitivity.csv`.

References

Bro, R., Acar, E., & Kolda, T. G. (2008). Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics*, *22*(2), 135–140. https://doi.org/10.1002/cem.1122

Davis, C., & Kahan, W. M. (1970). The Rotation of Eigenvectors by a Perturbation. III. *SIAM Journal on Numerical Analysis*, *7*(1), 1–46. https://doi.org/10.1137/0707001

Dimitrov, D., Knauer, C., Kriegel, K., & Rote, G. (2009). Bounds on the quality of the PCA bounding boxes. *Computational Geometry*, *42*(8), 772–789. https://doi.org/10.1016/j.comgeo.2008.02.007

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, *23*(1), 5–48. https://doi.org/10.1145/103162.103163

Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations* (4th ed.). Johns Hopkins University Press.

Gottschalk, S., Lin, M. C., & Manocha, D. (1996). OBBTree: A hierarchical structure for rapid interference detection. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, 171–180. https://doi.org/10.1145/237170.237244

Insight Software Consortium. (2026,). *itk::ShapeLabelObject Class Reference (Doxygen)*.

Lehmann, G. (2007). Label object representation and manipulation with ITK. *The Insight Journal*. https://doi.org/10.54294/q6auw4

O'Rourke, J. (1985). Finding minimal enclosing boxes. *International Journal of Computer and Information Sciences*, *14*(3), 183–199. https://doi.org/10.1007/BF00991005