Behavioral Invariants for Color Lookup Tables: Auditing What a Transform Does Rather Than How It Is Stored

Satyajeeth Suresh Kannan
Innovation Academy · Mary's Lab
github.com/JeetuSK0808/maryslab

Preprint · August 2026 · Mary's Lab technical report TR-11

ABSTRACT

Most shipped color transforms are 3D lookup tables. Existing tooling validates their structure — file well-formedness, lattice size, value range — and profile validators additionally check colorimetric accuracy against reference patches. None checks behaviour: whether the transform reverses tones along the neutral axis, tints greys, rotates hues past recognition, escapes range, or tears between adjacent cells. We define five such invariants and check them over every cell of a supplied lattice, reporting each violation with its lattice coordinates and a magnitude. On an identity lattice the audit reports no violations; on a synthetic "cinematic" look containing a deliberate tone kink it reports 1 tone reversals and 3 neutral contaminations. The audit takes the raw lattice rather than a file, so it applies to any format and to transforms sampled from a live pipeline.

Keywords: color management; lookup tables; software verification; invariants; color grading

1. Introduction

A creative look, a camera profile, a display calibration: each ships as a lattice of output triples indexed by input triples. The lattice is opaque. A reviewer can inspect a handful of patches, or render a test image and look, but neither procedure systematically covers the cells where defects hide.

The defects that matter are behavioural. A transform that reverses tone locally produces images where shadow and highlight relationships invert. A transform that tints neutrals shifts every grey surface in every frame. These are not aesthetic choices gone wrong; they are the transform doing something no sane transform should do, and they are detectable without knowing what the transform was supposed to look like.

2. Method

The audit accepts the raw lattice — $n^3 \times 3$ values in [0,1], red-fastest, the `.cube` convention — so it is format-independent and applies equally to a lattice sampled from a running pipeline. A lattice whose length does not match its declared size fails loudly rather than being reinterpreted.

Five invariants are checked:

- **Tone monotonicity.** Walking the neutral axis from black to white, perceived lightness must not decrease. A local inversion is a tone reversal.
- **Neutral preservation.** A grey input must not acquire chroma beyond a small threshold. Excess is a colour cast, and casts on neutrals are among the most visible defects.
- **Hue stability.** Colors chromatic enough for hue to be meaningful must not rotate beyond a threshold in Oklab.
- **Range discipline.** No output component outside [0,1]; escapes are clipped downstream, which shifts hue as well as level.
- **Local continuity.** Adjacent lattice cells must map to nearby outputs. An output jump many times the input step is a tear — usually corruption or a badly joined composition.

Every violation carries its check name, its exact lattice coordinates, and a magnitude, so a finding localises the defect rather than merely reporting one.

3. Evaluation

Inputs are synthetic and constructed with known ground truth. Two lattices at 9^3 resolution: an identity transform, which must produce no violations, and a "cinematic" look with a deliberately inserted tone kink over a narrow lightness band plus a warm–cool split. The second is a caricature of a real creative look, built so the defects it contains are known by construction.

Table 1. Violation counts by check, for an identity lattice and a deliberately broken look. The identity lattice is the control: any non-zero count would indicate a false positive in the audit itself.

check	identity	broken look
tone reversal	0	1
neutral contamination	0	3
hue flip	0	0
range escape	0	0
tear	0	14

The identity lattice produces no violations under any check, which is the control result: the audit does not manufacture findings on a transform that does nothing. The broken look is caught on the checks corresponding to the defects deliberately inserted, and crucially the findings carry coordinates, so the affected lightness band can be located directly rather than hunted for.

Two details in the broken column are worth stating rather than glossing. First, tears outnumber every other finding (14 against 1 tone reversal), and we did not insert tears deliberately. They are a consequence of the inserted defect: the kink switches on over a narrow lightness band, so every lattice edge crossing that band maps to a discontinuous jump. The tear check is detecting the same defect the tone check found, counted along the boundary of the affected region instead of along the neutral axis. This is worth knowing when reading counts — the checks are not independent, and a violation count is not a defect count.

Second, the hue and range checks report nothing, correctly: the constructed look clamps its output into [0,1] and applies only a mild warm–cool split, so there is nothing for those checks to find. A test set in which every check fires would be a weaker test set, since it could not distinguish a check that works from a check that always alarms.

The design point worth stating is that this is metamorphic testing applied to a data structure rather than a function. The invariants are properties any sane transform satisfies, checkable without a reference output — which is what makes the audit applicable to creative looks, where no ground truth exists by definition.

Reproducing these numbers

Clone `github.com/JeetuSK0808/maryslab` and run `node papers/build/experiments.mjs`. Every figure in this report is written by that script into `papers/build/results.json`; the instrument itself is exercised by `audit_color_transform`. The engines carry a 401-vector conformance suite shared by a TypeScript reference implementation, a C++20 core, and a WebAssembly build, so a reported number is a property of the specification rather than of one implementation.

4. Real-data evaluation: transforms that actually ship

The broken look was built to fail, which demonstrates that the checks fire but says nothing about whether they fire on things people use. The transforms below are not caricatures: they are the primaries conversions every colour-managed pipeline applies, and a contrast curve of the kind applied to footage daily. Unlike a creative look, these have published definitions, so what the audit says about them is checkable against the standards rather than against our intent.

Each was sampled onto the same 9^3 lattice the audit consumes, through the standard path — sRGB decode, matrix in linear light, re-encode.

Table 2. Violation counts for standardised transforms at 9^3 lattice resolution. Identity is the control.

transform	tone	neutral	hue	range	tear
identity	0	0	0	0	0
sRGB → Display-P3	0	0	0	0	0
sRGB → Rec.2020	0	0	0	0	0
per-channel S-curve	0	0	4	0	0

Both primaries conversions come back completely clean. That is the result we wanted and the one that is easy to undersell: a checker that flagged sRGB → Display-P3 would be worthless, because that transform is correct by construction and its correctness is not in question. Passing it is the audit demonstrating it can tell a well-formed transform from a malformed one, on transforms we did not write.

The per-channel S-curve is the interesting row. It raises 4 hue violations and nothing else — tone stays monotonic, neutrals stay neutral, nothing escapes range or tears. That pattern is diagnostic of a real and well-known property: applying the same contrast curve independently to R, G and B moves saturated colours in hue, because the three channels sit at different points on the curve and their ratios change. Colourists know this and

work around it by applying contrast in a luminance-preserving space. The audit recovers a known artifact of a standard operation, from the lattice alone, without being told what the transform was meant to do.

This is the strongest available evidence that the checks are measuring transform behaviour rather than our expectations of it. The finding was not planted; it is a consequence of how per-channel curves work, and the instrument found it.

Reproducing these numbers

Clone `github.com/JeetuSK0808/maryslab` and run `node papers/build/experiments.mjs`. Every figure in this report is written by that script into `papers/build/results.json`; the instrument itself is exercised by `audit_color_transform`. The engines carry a 401-vector conformance suite shared by a TypeScript reference implementation, a C++20 core, and a WebAssembly build, so a reported number is a property of the specification rather than of one implementation.

5. Limitations

This section states what the result does not establish. It is written to be read before the abstract is quoted.

- **A clean verdict on a standard transform is not a proof of the standard.** The primaries conversions pass these five checks at this lattice resolution. They are correct for reasons the audit does not verify, and a transform can satisfy every behavioural invariant here while implementing the wrong matrix.
- **Thresholds are engineering judgments.** The chroma bound for casts, the angular bound for hue flips, and the jump factor for tears are defensible defaults, not perceptual law. A deliberately stylised look will trip them on purpose, and the audit describes what a transform does rather than judging intent.
- **Only lattice points are observed.** Continuity is judged between adjacent cells, so a defect living strictly between the cells of a coarse lattice is invisible. The verdict describes the lattice supplied at the resolution supplied.
- **Reporting truncates.** A bounded number of violations is collected and returned; the per-check summary carries the counts.
- **Passing is not endorsement.** A transform with no violations does not reverse tones or tint neutrals. Whether it is suitable for a given footage is not a question this instrument answers.
- **Prior art was surveyed informally.** LUT validation and profile checking tools exist; a search found none testing behavioural invariants of the kind defined here.

6. Acknowledgements and disclosure

The instruments described here were implemented in Mary's Lab, an open-source computational color science project. Software design, implementation, and the preparation of this report were carried out with substantial assistance from a large language model (Anthropic Claude), under the author's direction and review. All numerical results were produced by executing the released code; none were generated by a language model. The author is responsible for the claims made here.

References

1. Ottosson, B. (2020). *A perceptual color space for image processing (Oklab)*. bottosson.github.io.
2. Chen, T. Y. et al. (2018). Metamorphic testing: a review of challenges and opportunities. *ACM Computing Surveys*, 51(1), 1–27.
3. Morovič, J. (2008). *Color Gamut Mapping*. Wiley.
4. CIE (2004). *Colorimetry, 3rd edition*. CIE 15:2004. Commission Internationale de l'Éclairage, Vienna.
5. International Color Consortium (2010). *Specification ICC.1:2010 — Image technology colour management*.
6. Selan, J. (2012). Cinematic color: from your monitor to the big screen. *ACM SIGGRAPH Courses*.
7. Sharma, G., Wu, W., and Dalal, E. N. (2005). The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1), 21–30.
8. Johnson, G. M. and Fairchild, M. D. (2003). A top down description of S-CIELAB and CIEDE2000. *Color Research & Application*, 28(6), 425–435.

Source, engines, and the script that produced every number in this report: `github.com/JeetuSK0808/maryslab`. Results regenerate with `node papers/build/experiments.mjs`. Inputs: synthetic where §3 says so, and measured USGS splib07a spectra (snapshot `usgs-splib07a-1`) where a real-data section says so. Prepared with AI assistance (see Acknowledgements).