Open Astro Core

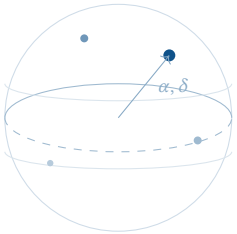
A Pure-Function SDK for Computational Astrophotography

Simon Knight

Adelaide, Australia

March 2026 • Version 0.12.0

Last updated: March 16, 2026



Abstract. Astrophotography demands a wide range of computational primitives—coordinate transforms, celestial mechanics, image statistics, noise modelling, frame registration, and stacking—yet existing tools scatter this logic across monolithic GUI applications with tightly coupled I/O. Open Astro Core (`astro-core`, `astro-vision`) is a Rust SDK that provides these primitives as pure, deterministic functions with no I/O or threading dependencies. This report describes the layered architecture, formalises the key algorithms (coordinate transforms, CMOS noise modelling, sigma-clipped stacking, triangle-similarity registration, and WCS with SIP distortion), and discusses the design rationale that enables the SDK to be embedded in contexts ranging from a Raspberry Pi edge daemon to an iOS framing assistant. At the time of writing the codebase comprises 64,800 lines of Rust across 185 source files with over 2,600 unit tests.

Contents

1 Introduction

Modern astrophotography is a computationally intensive discipline. A single deep-sky imaging session may require the operator to compute target visibility windows from celestial mechanics, derive optimal sub-exposure times from a camera noise model, calibrate raw frames against darks and flats, register sub-exposures by matching star patterns, and stack hundreds of frames with outlier-robust statistics.

Traditional tools—PixInsight, Siril, NINA—provide these capabilities, but their implementations are embedded inside monolithic applications with tightly coupled user interfaces and file I/O. This coupling makes it difficult to reuse individual algorithms in new contexts: an iOS framing assistant, a Raspberry Pi edge daemon, or a command-line automation pipeline.

Insight: *The astrophotography pipeline mirrors signal processing: acquire, calibrate, align, integrate, enhance. Each stage has well-defined mathematical foundations that benefit from isolation.*

Open Astro Core addresses this gap by providing a layered Rust Software Development Kit (SDK) in which every algorithm is a *pure function*: it takes typed inputs, produces typed outputs, and performs no I/O, no heap allocation beyond its return value, and no threading. This design yields three practical benefits:

1. **Embeddability.** The SDK compiles to a static library with no runtime dependencies, making it trivially embeddable via Foreign Function Interface (FFI) in Swift (iOS), Node.js (Electron), or Python (PyO3).
2. **Testability.** Every function is deterministic and can be tested with a simple assertion; no mocking of hardware, filesystems, or network connections is required.
3. **Composability.** Higher-level systems (e.g. a TUI orchestrator, an Instrument Neutral Distributed Interface (INDI) driver, or an HTTP API) compose SDK functions without inheriting implicit state or side effects.

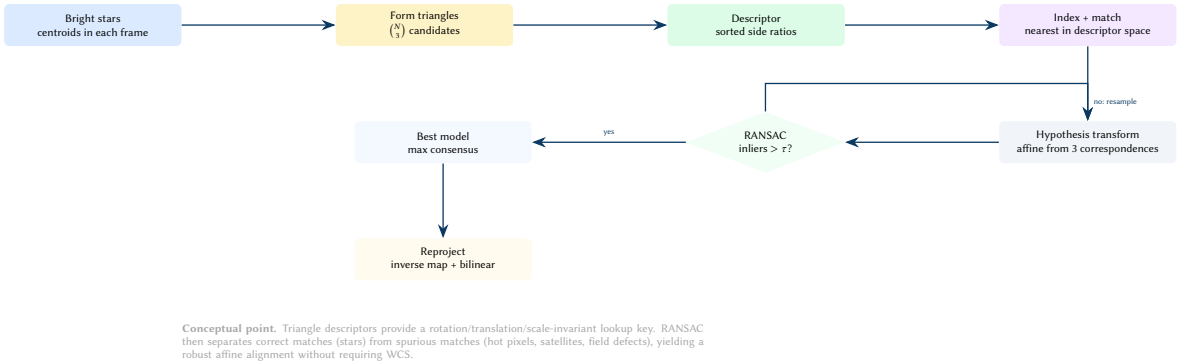


Figure 1. Triangle-similarity registration as a decision DAG: invariant descriptors propose correspondences; RANSAC selects a consensus transform; resampling reprojects the target frame.

This report makes the following contributions:

1. A description of the layered crate architecture and its dependency hierarchy (section 3).
2. A formalisation of the core coordinate transform pipeline, including sidereal time, equatorial-to-horizontal conversion, and atmospheric refraction (section 4).
3. A presentation of the imaging algorithms: Complementary Metal-Oxide-Semiconductor (CMOS) noise modelling, sigma-clipped stacking, triangle-similarity registration, and background estimation (section 5).
4. A description of the World Coordinate System (WCS) implementation with full Simple Imaging Polynomial (SIP) distortion polynomial support (section 6).
5. A discussion of the design rationale behind the pure-function constraint and its consequences for error handling and composability (section 7).

2 Background

Astrophotography processing involves a well-established pipeline: acquire, calibrate, register, stack, and stretch. The mathematical foundations draw from three domains.

2.1 Positional Astronomy

Positional astronomy provides the spherical trigonometry needed to convert between equatorial coordinates (right ascension, declination) and the observer's local horizon system (altitude, azimuth). The standard references are Meeus [0] for practical algorithms and the IAU SOFA library [0] for high-precision implementations. Open Astro Core follows the Meeus approach for its balance of accuracy and simplicity.

2.2 CMOS Noise Theory

The Signal-to-Noise Ratio (SNR) of a CMOS exposure is governed by the photon statistics of the signal and sky background, the sensor's read noise, and the dark current. The standard imaging sensor noise equation [0] is:

$$\mathrm{SNR} = \frac{S \cdot t}{\sqrt{S \cdot t + B_{\mathrm{sky}} \cdot t + D \cdot t + N \cdot \sigma_R^2}} \tag{1}$$

where S is the signal rate (e^-/s), t is the exposure time, B_{sky} is the sky background rate, D is the dark current, N is the number of stacked sub-exposures, and σ_R is the read noise in electrons. This equation underpins the exposure recommendation engine described in section 5.6.

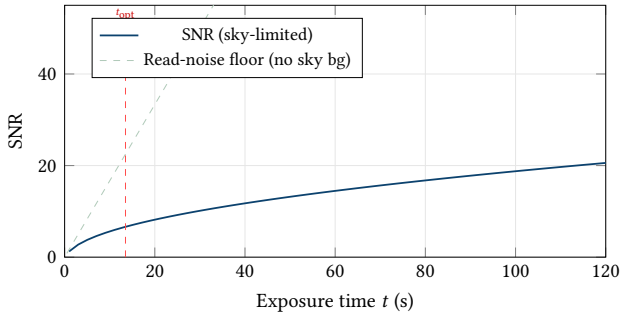


Figure 2. SNR as a function of sub-exposure time for a typical CMOS sensor. The dashed red line marks t_{opt}, the "swamp the read noise" threshold. Above this point the sky background noise dominates read noise and further extending the sub-exposure yields diminishing returns; the same total SNR can be achieved by stacking more shorter exposures.

2.3 Image Registration

Sub-exposure registration requires identifying corresponding stars between frames and computing a geometric transform that aligns them. Triangle-similarity matching [0] is the dominant approach in astronomical software: triangles formed by bright stars are described by their sorted side ratios, which are invariant to translation, rotation, and uniform scaling. Open Astro Core implements this algorithm with a RANSAC-style [0] consensus step, described in section 5.7.

3 Architecture

Open Astro Core is organised as a Cargo workspace containing eleven crates. The crates form a strict dependency hierarchy: lower layers provide pure computation; upper layers add I/O, hardware interaction, and orchestration.

3.1 Crate Hierarchy

Figure 3 illustrates the dependency graph. The two foundation crates—`astro-core` and `astro-vision`—contain all algorithms discussed in this report. They have no dependencies on tokio, device drivers, or file I/O libraries beyond the Flexible Image Transport System (FITS) and RAW format parsers in `astro-vision`.

Insight:
The "inverted pyramid" ensures that the most-reused code (math primitives) has zero dependencies beyond `serde` and `thiserror`.

: Dependency Inversion

Foundation crates (`astro-core`, `astro-vision`) must never depend on I/O, async runtimes, or device-specific code. All hardware interaction is mediated through traits defined in the foundation and implemented in upper-layer crates.

3.2 Foundation: astro-core

The foundation crate providing coordinate math, celestial mechanics, and planning primitives. Key modules include:

- `angle`, `sexagesimal` — angle newtypes with wrapping arithmetic and HMS/DMS parsing.
- `transforms` — equatorial↔horizontal conversion, angular separation (haversine), atmospheric refraction (Bennett's formula).
- `time` — Julian Date, Greenwich Mean Sidereal Time (GMST), Local Sidereal Time (LST) computation.
- `visibility` — airmass (Pickering), rise/transit/set, altitude curves.
- `solar`, `lunar` — Meeus-derived solar and lunar ephemeris with eclipse prediction.
- `session` — imaging window scoring, parallactic angle, field rotation rate, max untrailed exposure.

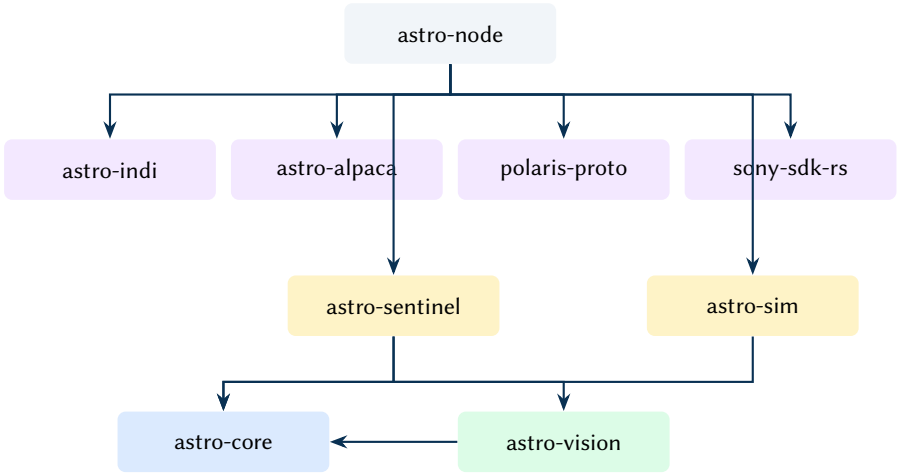


Figure 3. Crate dependency hierarchy. Arrows point from dependant to dependency. The two crates at the base contain all pure computation; upper layers add I/O and device interaction.

- `wcs` — WCS with TAN projection and SIP distortion polynomials (section 6).
- `mosaic` — mosaic grid generation with RA wrapping, serpentine ordering, and coverage tracking.
- `guiding` — co-axial offset calibration, guide quality analysis (RMS, periodic error FFT), spiral/grid search patterns.

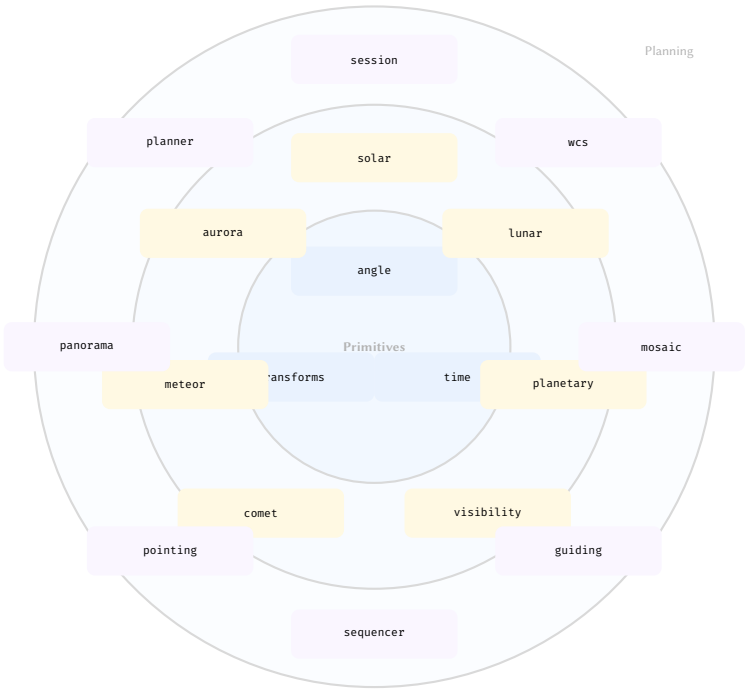


Figure 4. Module topology of `astro-core` organised in concentric rings. Inner modules (angle, time, transforms) are pure primitives with no intra-crate dependencies. Middle modules implement celestial mechanics. Outer modules provide planning and orchestration logic.

3.3 Foundation: astro-vision

Image processing algorithms operating on `Array2<f32>` from the `ndarray` crate. All pixel data is normalised to the $[0, 1]$ range.

- `stats` — median, MAD, sigma-clipped statistics, histogram, centroid.
- `background` — tiled sigma-clipped background estimation with bilinear interpolation.
- `stardetect` — threshold-and-flood-fill star detection with Half-Flux Radius (HFR) and Full Width at Half Maximum (FWHM) measurement.
- `stacking` — mean, median, sigma-clipped mean, and Winsorized sigma clipping.
- `registration` — triangle-similarity star matching with RANSAC affine fitting.
- `calibrate` — dark/flat/bias calibration with hot/cold pixel detection and interpolation.

- `noise_model` — CMOS noise model with SNR computation and optimal sub-exposure derivation.
- `exposure` — exposure recommendation, saturation prediction, Holy Grail ramp for timelapse.
- `autofocus` — V-curve fitting with IRLS Huber weights.
- `fits_write`, `ser` — FITS and Simple Extended Recording (SER) file I/O.
- `stretch` — Midtone Transfer Function (MTF) auto-stretch.

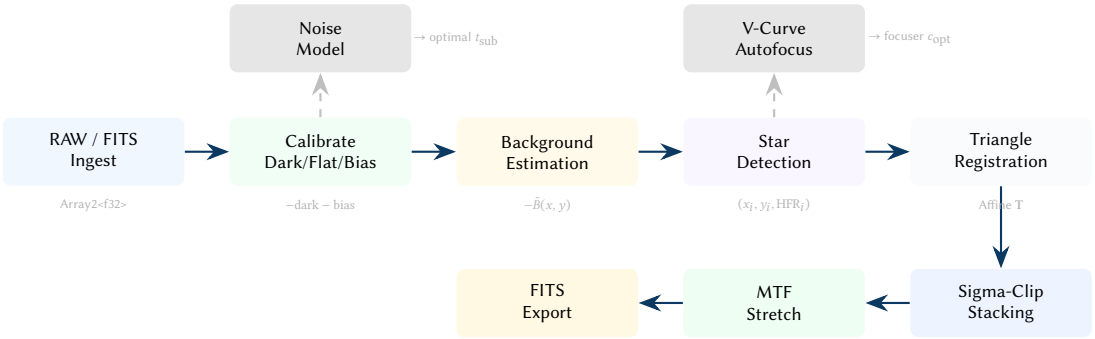


Figure 5. The `astro-vision` image processing pipeline. Data flows left-to-right through calibration, background subtraction, star detection, and registration before stacking. Side branches feed autofocus and noise modelling.

3.4 Mid-Tier Crates

3.4.1 astro-sentinel

A multi-stage safety pipeline that protects equipment and classifies sky conditions in real time. The pipeline operates in three phases: (1) *preprocessing*, where raw camera frames are downsampled and normalized for inference; (2) *classification*, where an ONNX neural network (or the built-in IR thermal classifier) produces labeled detections with confidence scores; and (3) *rule evaluation*, where a TOML-serializable rule set maps classifications to trigger actions (burst capture, notification, or emergency shutdown). The pipeline produces structured `Explanation` objects that record which rules matched, their confidence scores, and the resulting actions—providing a full audit trail for post-session analysis. The sentinel integrates directly with the sequencer: a safety state change triggers an immediate event that can park the mount and close the observatory within seconds.

3.4.2 astro-sim

A hardware-in-the-loop (HITL) virtual observatory that simulates the complete imaging chain without physical equipment. The mount model includes slew kinematics with acceleration profiles, periodic error injection (sinusoidal PE with configurable amplitude and period), and German equatorial meridian flip logic with configurable flip offset. The camera simulator renders star fields by projecting catalog positions through the optical system's plate scale, convolving with a Gaussian PSF, injecting Poisson shot noise and read noise, and optionally overlaying OpenSimplex cloud layers. Multi-device axis offsets model the real-world misalignment between guide scope and imaging scope. The simulator enables end-to-end integration testing of the entire sequencer–guider–vision pipeline, including edge cases like meridian flip mid-exposure, cloud interruption recovery, and power budget exhaustion—scenarios that are difficult to reproduce on demand with real hardware.

3.5 Driver Crates

`astro-indi` and `astro-alpaca` implement the two dominant telescope control protocols (INDI and Astronomy Common Object Model (ASCOM) Alpaca). `polaris-proto` is a native binary protocol driver for the Benro Polaris star tracker. `sony-sdk-rs` provides safe Rust bindings over the Sony Camera Remote SDK via a C ABI wrapper (`sony-sdk-sys`).

4 Celestial Mechanics

This section describes the core positional astronomy pipeline: sidereal time, coordinate transforms, and atmospheric refraction.

4.1 Sidereal Time

The GMST converts a Julian Date JD to the hour angle of the vernal equinox. Open Astro Core uses the Meeus cubic approximation [0]:

$$\Theta_{\mathrm{GMST}} = 280.46061837 + 360.98564736629\,(JD - 2451545.0) + 0.000388\,T^2 - \frac{T^3}{38710000} \tag{2}$$

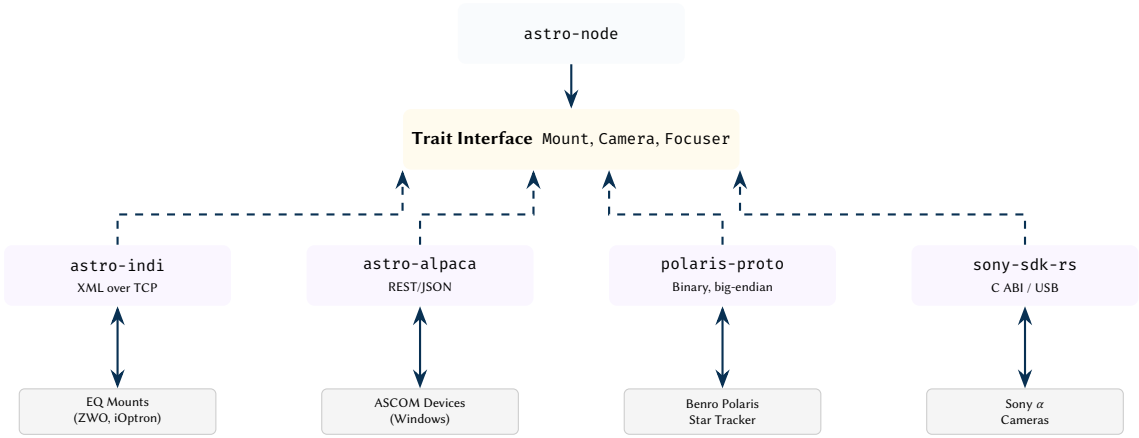


Figure 6. Driver crate protocol stack. Each driver implements a common trait interface, allowing `astro-node` to orchestrate heterogeneous hardware through a unified API regardless of the underlying wire protocol.

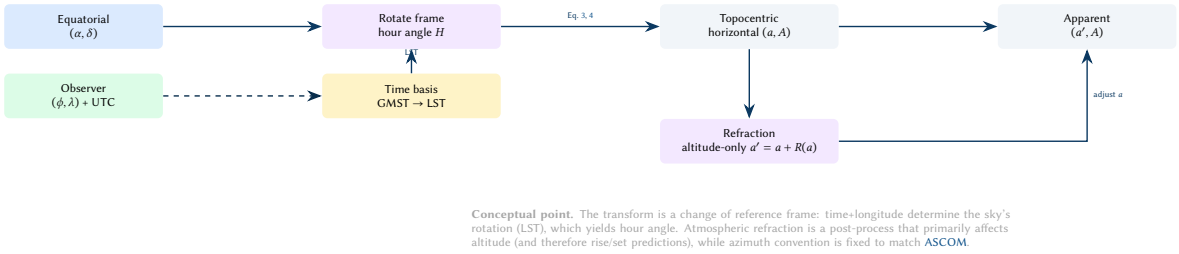


Figure 7. Coordinate transforms as frame changes with explicit invariants. Solid arrows carry coordinate transforms; dashed arrows carry time-basis inputs.

where $T = (JD - 2451545.0)/36525$ is the Julian century. LST is then LST = GMST + $\lambda/15$ where λ is the observer's east longitude in degrees.

> **Design Decision 1: Why Meeus Over SOFA?**
>
> The IAU SOFA library provides sub-milliarcsecond precision through nutation and precession corrections. For astrophotography—where the dominant error sources are polar alignment (~1 arcmin), atmospheric refraction (~30 arcsec), and mount periodic error (~10 arcsec)—the Meeus approximation's ~1 arcsec accuracy is more than sufficient, and the resulting code is self-contained with no external C dependencies.

4.2 Equatorial-to-Horizontal Transform

Given equatorial coordinates (H, δ) where H is the hour angle and δ is the declination, the standard spherical trigonometry yields the altitude a and azimuth A:

$$\sin a = \sin\phi \, \sin\delta + \cos\phi \, \cos\delta \, \cos H \tag{3}$$

$$A = \operatorname{atan2}(-\cos\delta \, \sin H, \;\; \sin\delta \, \cos\phi - \cos\delta \, \cos H \, \sin\phi) \tag{4}$$

where ϕ is the observer's latitude. The azimuth follows the ASCOM convention: North = 0, East-positive (clockwise).

Insight: *The North=0 East-positive convention differs from the mathematical standard (East=0 counter-clockwise). Matching ASCOM avoids a conversion step in every driver.*

The implementation clamps the $\sin a$ argument to $[-1, 1]$ to prevent `NaN` from floating-point rounding at the zenith and nadir.

The implementation follows Meeus Chapter 13 exactly, with a defensive `clamp` of $\sin a$ to $[-1, 1]$ before the arcsine call to guard against floating-point overshoot at the zenith and nadir. Azimuth is computed via the two-argument arctangent $A = \operatorname{atan2}(-\cos\delta \sin h, \; \sin\delta \cos\phi - \cos\delta \cos h \sin\phi)$ and wrapped to $[0^\circ, 360^\circ)$.

4.3 Angular Separation

The angular distance between two points on the celestial sphere uses the haversine formula for numerical stability at small separations:

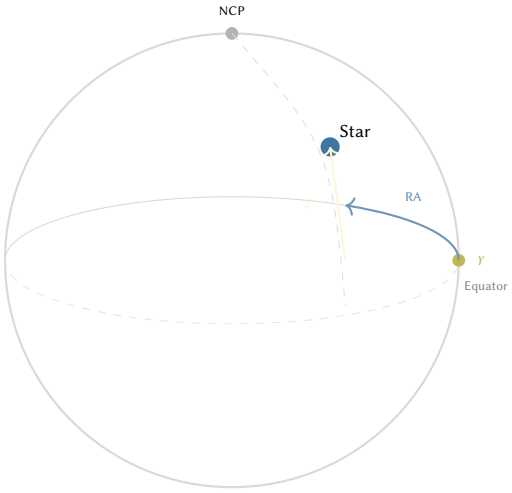


Figure 8. The equatorial coordinate system. Right Ascension (α) is measured eastward along the celestial equator from the vernal equinox (γ). Declination (δ) is measured north/south from the equator along the hour circle through the object.

$$d = 2\,\arcsin\sqrt{\sin^2\tfrac{\Delta\delta}{2} + \cos\delta_1\,\cos\delta_2\,\sin^2\tfrac{\Delta\alpha}{2}} \tag{5}$$

The naive dot-product formula $d = \arccos(\mathbf{u}\cdot\mathbf{v})$ loses precision below ~0.1 arcsec due to the flat gradient of arccos near unity. Since astrophotography routinely measures separations of a few arcseconds (e.g. guide star offsets, plate-solve residuals), the haversine form is essential.

4.4 Atmospheric Refraction

The apparent altitude of a celestial object is increased by atmospheric refraction. Open Astro Core applies Bennett's formula [0]:

$$R = \frac{1}{\tan\!\left(h + \frac{7.31}{h+4.4}\right)} \quad \text{(arcminutes)} \tag{6}$$

with a multiplicative temperature/pressure correction factor. This correction is applied in the rise/transit/set calculator, where it shifts the effective horizon from 0° to approximately $0^\circ 34'$.

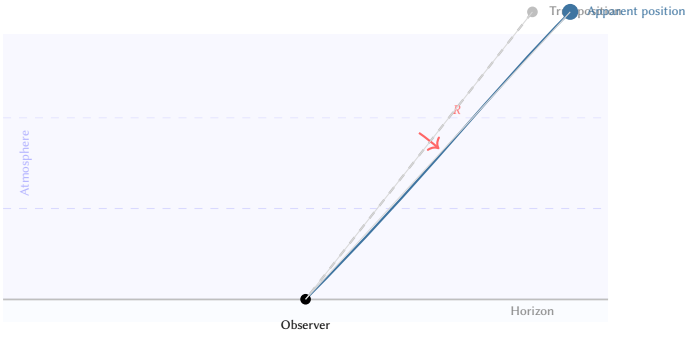


Figure 9. Atmospheric refraction bends starlight toward the zenith, making objects appear higher than their true geometric position. The effect is strongest near the horizon ($R \approx 34'$) and negligible at the zenith ($R < 0.1'$).

4.5 Airmass

The airmass X quantifies the path length of light through the atmosphere relative to the zenith. Open Astro Core uses Pickering's improved formula [0], which remains accurate down to the horizon:

$$X = \frac{1}{\sin\!\left(h + \frac{244}{165+47\,h^{1.1}}\right)} \tag{7}$$

where h is the true altitude in degrees. The standard $\sec z$ approximation diverges below ~10° altitude; Pickering's formula agrees with rigorous ray-tracing models [0] to within 0.1% at all altitudes above 1°.

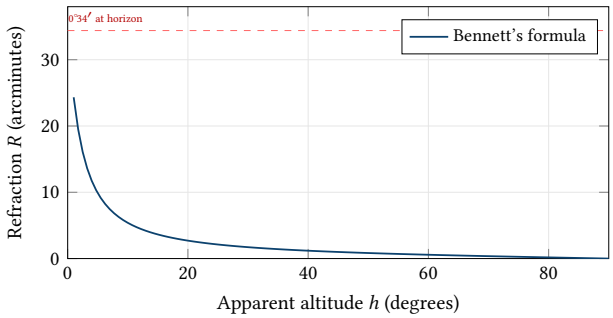


Figure 10. Atmospheric refraction magnitude as a function of apparent altitude using Bennett's formula. Refraction reaches ~34′ at the geometric horizon, causing objects to appear 0.57° higher than their true position. Above 20° the correction falls below 3′ and is often negligible for wide-field framing, but remains significant for rise/set time predictions.

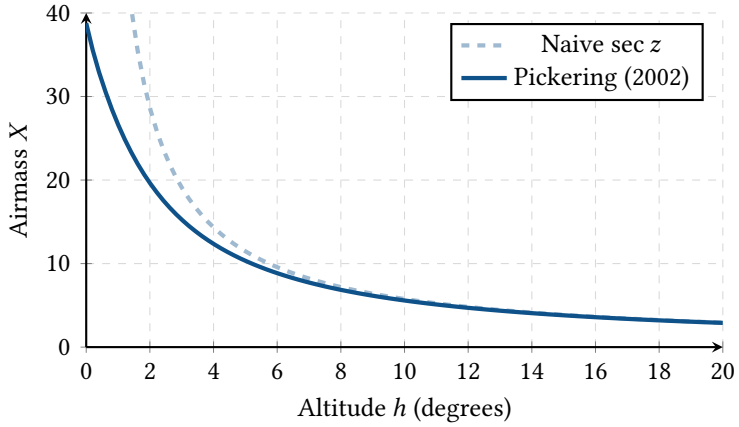


Figure 11. Comparison of airmass models near the horizon. The naive secant approximation diverges to infinity at $h = 0°$, whereas Pickering's interpolation correctly models atmospheric curvature to match empirical scattering data ($X \approx 38$ at the horizon).

4.6 Lunar Ephemeris

The `lunar` module implements Meeus Chapter 47 with 10+ perturbation terms for ecliptic longitude and 6+ for latitude. The geocentric ecliptic position is converted to equatorial via the obliquity of the ecliptic, then corrected for topocentric parallax using the observer's geocentric latitude and distance.

Key derived quantities include:

- **Phase illumination** from the elongation angle D, accounting for perturbation corrections.
- **Eclipse prediction** via umbral/penumbral shadow geometry, with binary search for maximum eclipse and contact-time detection by stepping.

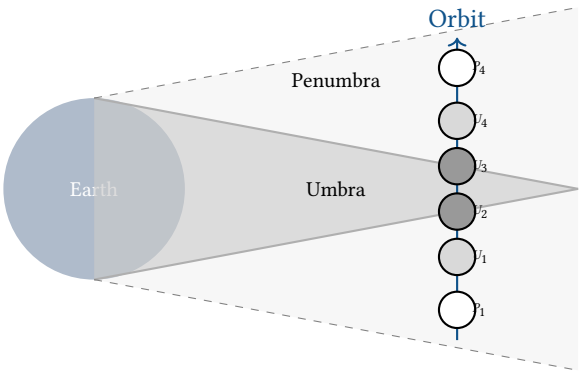


Figure 12. Lunar eclipse shadow geometry showing the Earth's umbral and penumbral cones. The SDK predicts contact points (P_1, U_1, U_2, etc.) by stepping through the Moon's geocentric ecliptic coordinates.

4.7 Polar Alignment Error & Drift

The `polar_align` module mathematically models the drift of a star caused by the misalignment between the mount's mechanical right ascension axis and the True Celestial Pole (TCP).

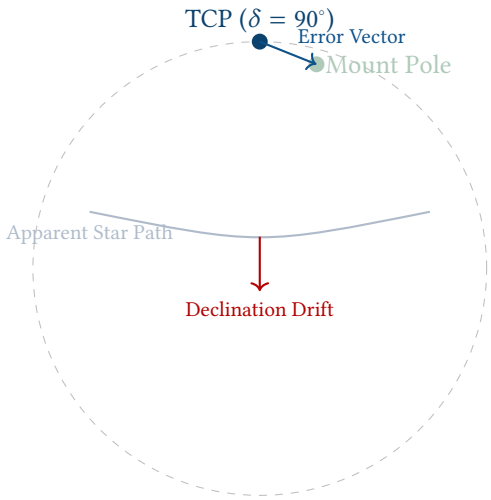


Figure 13. Polar alignment error. A misalignment between the True Celestial Pole (TCP) and the mount's mechanical polar axis causes stars to exhibit a continuous declination drift over time, which forms the basis of the King drift method.

By tracking a star's apparent declination drift over time without guiding, the system solves the spherical trigonometry required to compute the exact azimuth and altitude offset vectors needed to physical align the mount, formalizing the traditional King Drift Method [0].

4.8 Target Visibility & Horizon Masks

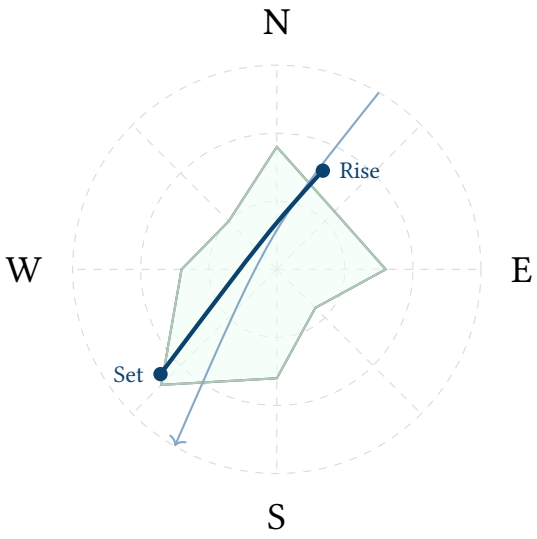


Figure 14. Altitude-Azimuth polar projection (zenith at center, horizon at edge). The shaded polygon represents the local topographical horizon mask (e.g., trees, buildings). The target's nightly arc intersects the mask to compute precise visibility windows.

The `horizon_mask` module models the local observing topology using a discrete altitude-azimuth polygon. Instead of assuming a flat 0° horizon, the planner intersects the celestial target's computed daily altitude curve with the local mask polygon to derive true unoccluded observing windows, preventing the scheduler from tracking targets behind trees or buildings.

4.9 Target Planning & Session Scoring

Choosing the right target for a given night requires combining multiple quality signals into a single actionable recommendation. The `session` module provides two complementary tools for this.

4.9.1 Twilight Boundaries

The `twilight_times` function bisection-searches the Sun's altitude curve to locate the six twilight boundaries—civil, nautical, and astronomical—for both dusk and dawn. This defines the usable imaging window for the night.

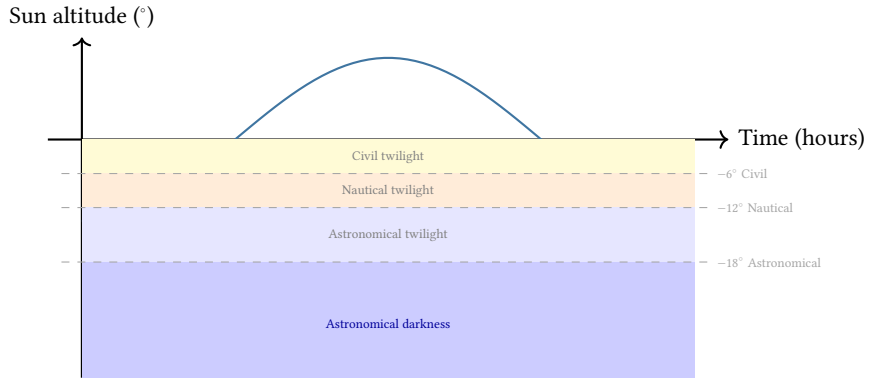


Figure 15. Twilight boundary model. The `twilight_times` function locates the six solar altitude crossings that delimit civil, nautical, and astronomical twilight, bounding the usable deep-sky imaging window.

4.9.2 Imaging Window Score

The `imaging_window_score` function collapses four orthogonal quality signals into a single $[0, 1]$ score at a given Julian Date:

$$Q = Q_{\text{alt}} \cdot Q_{\text{air}} \cdot Q_{\text{moon}} \cdot Q_{\text{dark}} \tag{8}$$

where:

- Q_{alt}: altitude factor—zero below horizon, linear from 0° to 30°, unity above 30°.
- Q_{air}: airmass factor—$1 - (\mathcal{X} - 1)/3$, clamped to $[0, 1]$. Penalises low-elevation targets whose light traverses more atmosphere.
- Q_{moon}: moon proximity factor—zero within 0°, unity beyond 60° angular separation, linear between.
- Q_{dark}: darkness factor—unity in astronomical darkness ($\odot < -18°$), 0.5 in nautical twilight, zero otherwise.

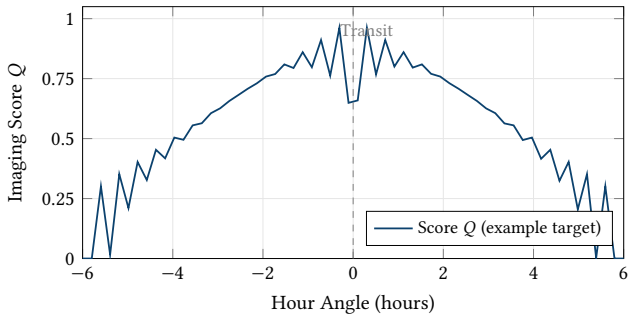


Figure 16. Imaging window score over a night for a representative target at declination +45°. The score peaks near transit (hour angle $H = 0$) where the target reaches maximum altitude and minimum airmass, then decays symmetrically as the target descends toward the horizon.

> **Design Decision 2: Multiplicative Score Design**
>
> Multiplying four factors—rather than summing weighted components—ensures that a single disqualifying condition (below horizon, daytime, full Moon on target) collapses the score to zero regardless of how favourable the other signals are. This prevents the scheduler from selecting a technically high-altitude target that is flooded by moonlight.

4.9.3 Multi-Factor Target Scoring

While the instantaneous imaging-window score Q (eq. (8)) captures the quality of a single moment, the `planner` module integrates across the entire observing window to produce a single actionable target recommendation. Given a candidate list $\{T_i\}$, a `PlannerWindow` $[t_0, t_1]$ sampled at Δt intervals, and a `PlannerSite` that encodes

observer location, Bortle class, horizon mask, filter profile, and battery state, each target receives a composite score:

$$S = w_Q \bar{Q} + w_t f_t + w_m \hat{m} + w_k \cdot k \cdot \phi + w_p \cdot p \tag{9}$$

where:

- $\bar{Q}$: Mean imaging-window score across all samples in the window.
- f_t: Fraction of samples where target altitude exceeds the effective minimum (the pointwise maximum of the global `min_alt_deg` and the local horizon mask at the target's azimuth).
- $\hat{m}$: Minimum Moon-target angular separation during the window, normalised to 90°, i.e. $\hat{m} = \min(\theta_{\text{moon}})/90$.
- k: A Bortle-modulated target-kind preference (see below).
- ϕ: A filter–target synergy multiplier (see below).
- p: Predictive power feasibility—the ratio of estimated remaining safe battery hours to the block duration, clamped to $[0, 1]$.

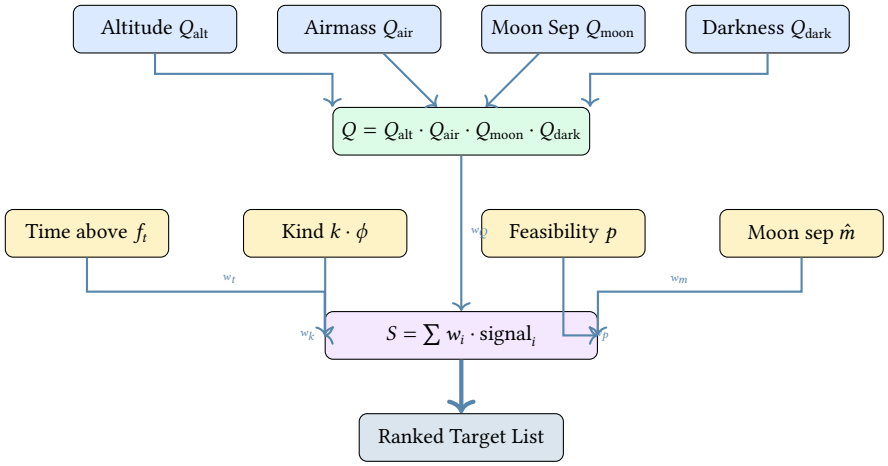


Figure 17. Multi-factor target scoring pipeline. Four instantaneous quality signals are multiplied into a single imaging-window score Q, then integrated across the planning window alongside time-above-horizon, target–filter synergy, Moon separation, and battery feasibility to produce a single composite score S per target.

The default weight vector is $(w_Q, w_t, w_m, w_k, w_p) = (0.50, 0.20, 0.15, 0.05, 0.10)$, but the caller may supply custom `PlannerWeights` to adjust priorities—for example, a remote robotic observatory might increase w_p to emphasise power conservation.

> **Design Decision 3: Additive vs. Multiplicative Aggregation**
>
> The top-level planner score S is an *additive* weighted sum, unlike the multiplicative instantaneous score Q. This is intentional: the multiplicative Q enforces hard vetoes (below horizon ⇒ zero), while the additive S allows trade-offs at the planning level—a target with slightly less Moon separation but significantly more hours above the horizon may still rank higher. The veto semantics are preserved because $\bar{Q}$ itself collapses to zero for disqualified targets, zeroing the dominant w_Q term.

4.9.4 Bortle-Aware Target–Kind Preference

Not all deep-sky objects are equally rewarding under all sky conditions. The planner assigns a *kind preference* factor $k \in [0, 1]$ that modulates the score based on the object type and the site's Bortle class:

The rationale is straightforward: galaxies are broadband-continuum objects whose faint outer arms are obliterated by light pollution, whereas emission nebulae radiate primarily in discrete spectral lines (Hα at 656.3 nm, O-III at 500.7 nm, S-II at 672.4 nm) that can be isolated with narrowband filters even under Bortle 8 suburban skies.

4.9.5 Filter–Target Synergy

The kind preference k is further modulated by a *synergy multiplier* ϕ that captures the interaction between the observer's currently selected filter profile and the target type. This accounts for the fact that shooting a galaxy through a narrowband Hα filter yields poor results regardless of sky conditions.

The combined $k \cdot \phi$ product allows the planner to make nuanced recommendations: under Bortle 7 skies with an Hα filter mounted, emission nebulae receive $k = 1.0$ and $\phi = 1.15$, while a galaxy receives $k = 0.4$ and

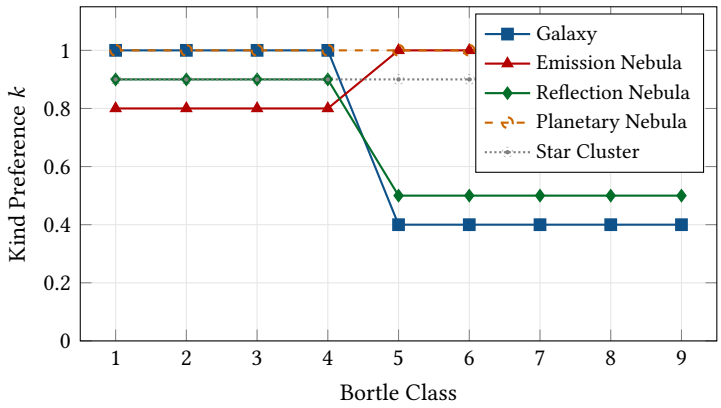


Figure 18. Bortle-modulated target-kind preference. Under dark skies (B1–B4), broadband-sensitive objects like galaxies and reflection nebulae are prioritised. Under light-polluted skies (B5–B9), emission nebulae—which respond well to narrowband filtration—receive the highest preference. Planetary nebulae and star clusters are relatively site-agnostic.

	Broadband	**Hα**	**O-III**	**S-II**
Galaxy	**1.10** / 0.60	0.85	0.85	0.85
Emission	1.00 / 0.90	**1.15**	1.00	**1.10**
Reflection	**1.00** / 0.70	0.85	0.85	0.85
Plan. Neb.	1.00	**1.10**	1.05	1.00
Cluster	0.95	0.95	0.95	0.95

Broadband column: dark sky / bright sky (B6+). Bold = optimal pairing.

Figure 19. Filter–target synergy matrix ϕ. The strongest synergy ($\phi = 1.15$) pairs emission nebulae with Hα; the weakest ($\phi = 0.60$) pairs galaxies with broadband under bright skies. Broadband values are shown as dark-sky / bright-sky pairs.

$\phi = 0.85$—a 3.4× scoring advantage for the emission target, correctly reflecting real-world astrophotography priorities.

4.9.6 Contiguous Block Extraction

Knowing that a target is "good tonight" is insufficient for schedule construction—the planner must identify the *best contiguous imaging block* within the window. The `compute_best_block` function implements a sliding-window search:

1. Compute the block length $L = \lceil t_{\min}/\Delta t \rceil$ where $t_{\min}$ is the configurable minimum block duration (default 60 minutes).
2. Slide a window of L samples across the time series.
3. Reject any window where any sample has altitude below the effective minimum.
4. Among valid windows, select the one with the highest mean imaging-window score $\bar{Q}_{\text{block}}$.
5. Return the `SuggestedBlock`: start/end JD, average score, minimum Moon separation, and duration in hours.

4.9.7 Schedule Timeline Construction

Once the best block is identified, the `plan_schedule_for_block` function constructs a fully-tiled, monotonic, non-overlapping schedule timeline for the entire planning window. The timeline includes:

1. **Leading Gap**—idle time from window start to imaging overheads.
2. **Slew**—mount slew to target (default 30 s).
3. **Settle**—mount settle time (default 5 s).
4. **Imaging segments**—broken by periodic focus checks.
5. **Focus Checks**—periodic autofocus runs (default every 30 minutes, 60 s duration each).
6. **Trailing Gap**—post-imaging idle time to window end.

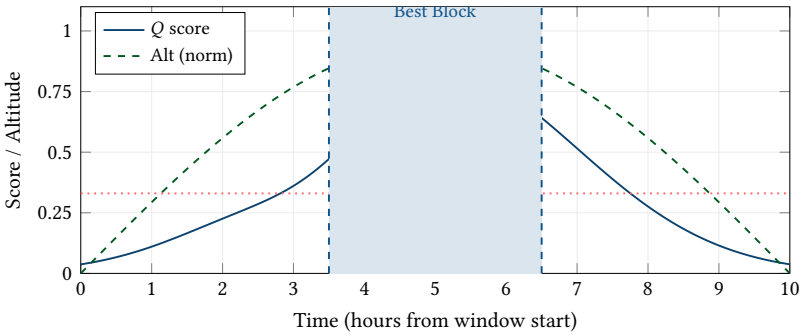


Figure 20. Contiguous block extraction. The sliding window identifies the 3-hour segment (shaded) with the highest mean Q score while ensuring that the target remains above the minimum altitude threshold (dotted line) throughout the block.

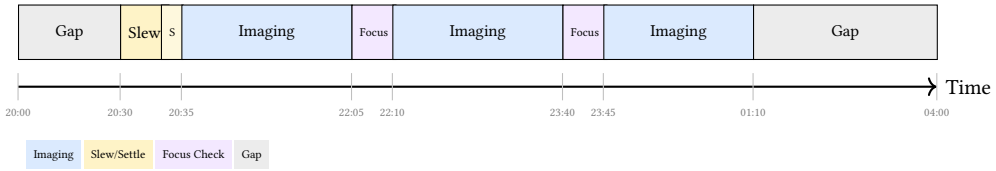


Figure 21. Example schedule timeline for a single-target night. Imaging segments are interleaved with periodic focus checks. The leading and trailing gaps account for twilight boundaries and target rise/set times.

4.9.8 Moon Phase & Lunation Calendar

The lunar phase is the single most impactful environmental factor for deep-sky astrophotography. A full Moon raises the sky background by ~2 magnitudes per square arcsecond, overwhelming faint nebulosity and galactic detail. The `lunar` module computes the phase via:

$$k = \frac{1 + \cos i}{2} \tag{10}$$

where i is the selenocentric elongation angle, computed from Meeus-based perturbation series on the lunar mean anomaly M', solar mean anomaly M, and mean elongation D. The illumination fraction k ranges from 0 (new Moon) to 1 (full Moon).

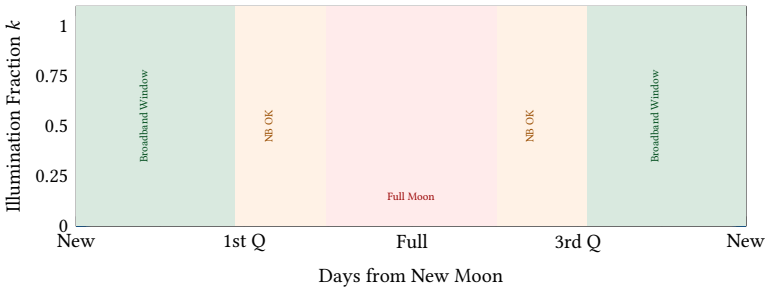


Figure 22. Lunation-based observing strategy. The green zones around new Moon are optimal for broadband imaging; the orange zones tolerate narrowband filtration; the red zone near full Moon is unsuitable for deep-sky imaging without aggressive narrowband filters.

The planner uses the Moon's position and illumination in two ways: first, the instantaneous Moon-target angular separation enters Q via Q_{moon}; second, the minimum separation during the planning window enters the composite score S via the $\hat{m}$ term. Together these ensure that targets near the Moon—regardless of how high they transit—are penalised appropriately.

4.9.9 Multi-Night & Seasonal Opportunity Windows

A single-night planner answers "what should I image tonight?", but serious astrophotographers plan projects that span weeks or months. The `planner` scoring engine supports multi-night planning by accepting arbitrary `PlannerWindow` extents. When the window spans multiple nights, the integrated $\bar{Q}$ naturally accounts for the lunation cycle, seasonal declination drift, and twilight duration changes.

By scoring the same target list across weekly windows for an entire season, the planner produces a *seasonal opportunity matrix* that reveals when each target peaks. This enables front-end applications to present

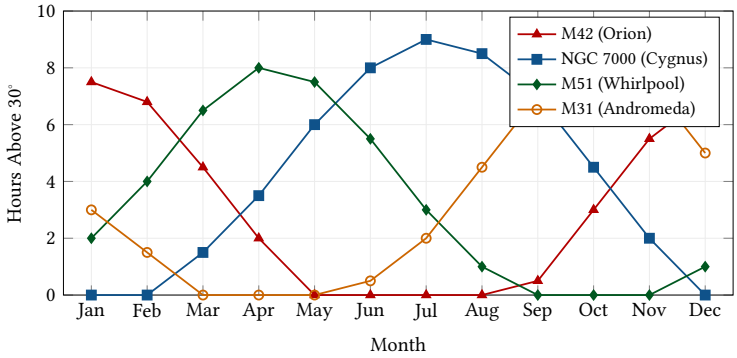


Figure 23. Seasonal visibility windows for representative deep-sky targets at mid-northern latitudes ($\phi = 35°$). Each target has a well-defined "season" of 3–5 months where it transits at useful altitudes during astronomical darkness. The planner's time-above-threshold metric f_t captures this seasonality automatically.

"start imaging M42 in October, peak in December" guidance without hard-coding any astronomical knowledge.

4.9.10 Equipment-Aware Sub-Frame Budget

The planning layer connects target selection to the imaging pipeline via the `exposure` and `noise_model` modules. Given a `CameraProfile` (gain, read noise, dark current, full-well capacity), a Bortle-derived sky background estimate, and the target's expected signal rate, the system computes:

1. The *optimal sub-exposure time* t_{opt} from the sky-noise swamping criterion:

$$t_{\text{opt}} = \frac{(r \cdot \sigma_r)^2}{B_{\text{sky}}} \tag{11}$$

   where $r = 3$ ensures that sky noise exceeds read noise by 3×, making the read noise contribution negligible (~10%).
2. The *total number of sub-frames* $n = T_{\text{total}}/t_{\text{opt}}$ given the available imaging hours from the suggested block.
3. The predicted *stack SNR* from the standard CCD equation:

$$\text{SNR} = \frac{S \cdot t \cdot n}{\sqrt{n \cdot (S \cdot t + B_{\text{sky}} \cdot t + D \cdot t + \sigma_r^2)}} \tag{12}$$

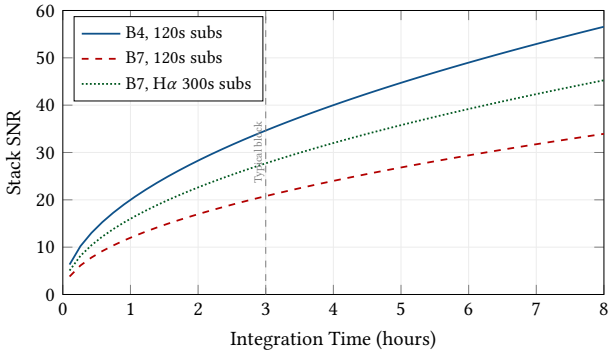


Figure 24. Predicted stack SNR versus integration time for different equipment–site combinations. Under dark skies (B4), broadband imaging achieves higher SNR per hour; under light-polluted skies (B7), narrowband Hα partially compensates by rejecting the elevated sky background. The dashed line marks a typical 3-hour imaging block.

This tight coupling between the planner and the exposure engine allows the UI to present not just "image M42 tonight" but "image M42 tonight with 90×120 s Hα subs for a predicted stack SNR of 38."

4.9.11 Sub-Frame Progress Tracking

For multi-session projects, the planner must account for sub-frames already captured. The scoring framework supports a *completion-aware* mode where the caller supplies the number of subs already acquired per filter channel. The remaining integration deficit is then used to compute a priority boost: a target needing only 20 more minutes of Hα to complete a panel ranks higher than one requiring a full 6-hour broadband campaign, enabling the planner to recommend "finish this, then start that" strategies.

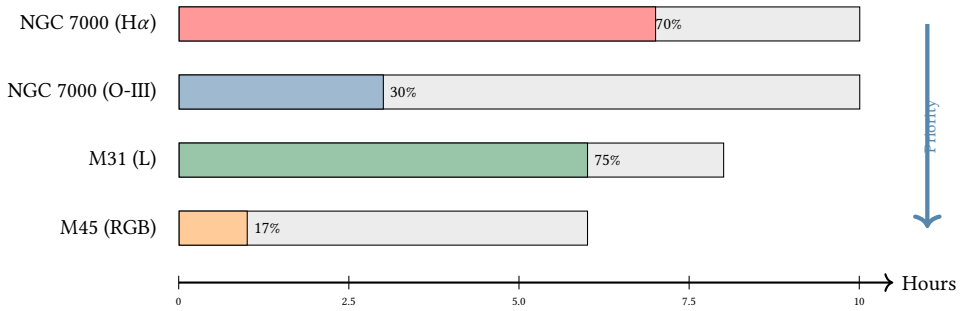


Figure 25. Sub-frame progress tracking across a multi-target project. Targets closer to completion receive a priority boost (top), enabling the planner to recommend finishing near-complete channels before starting new ones.

4.9.12 Advanced Narrowband Planning

Narrowband imaging presents unique planning challenges. The standard SHO (Sulfur-II, Hydrogen-alpha, Oxygen-III) palette requires three separate filter channels, each with different optimal conditions and dramatically different exposure requirements.

The `filter` module provides two functions for narrowband campaign management:

Filter ranking. The `rank_filters_for_target` function scores each available filter against the current target type, Bortle class, and Moon illumination fraction. Key heuristics include:

- Narrowband filters receive a +1.5 bonus when the Moon illumination exceeds 60%, reflecting their ability to reject scattered moonlight.
- Narrowband filters receive a +1.5 bonus under Bortle ≥ 6 skies, reflecting their immunity to broadband light pollution.
- Broadband luminance receives a +1.0 bonus under dark skies ($B < 5$) with low Moon, where it captures the widest dynamic range.

Wheel rotation optimisation. The `optimize_filter_sequence` function minimises the total physical rotation distance of the filter wheel by sorting filters ascending by slot number. While conceptually simple, this reduces mechanical wear and eliminates backlash-induced focus shifts on friction-drive filter wheels.

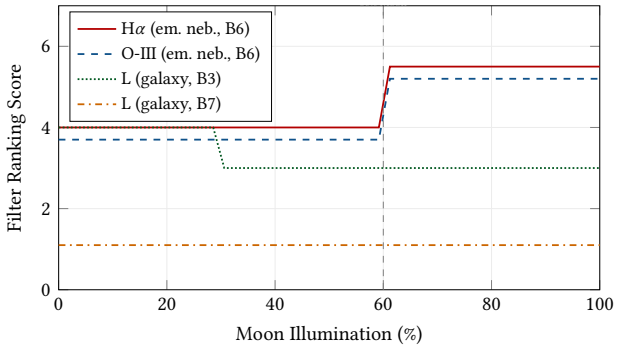


Figure 26. Filter ranking score as a function of Moon illumination. Narrowband filters (Hα, O-III) receive a step-function bonus above 60% illumination, making them the clear recommendation for sessions near the full Moon. Broadband luminance is only competitive under dark, moonless conditions.

> **Design Decision 4: Narrowband Under Moonlight**
>
> A common misconception among novice astrophotographers is that imaging is impossible near the full Moon. In practice, 3–7 nm narrowband filters reject >99.5% of scattered moonlight, enabling productive Hα and O-III acquisition even at 90% illumination. The planner's Moon-aware filter ranking codifies this domain expertise, automatically switching recommendations from broadband to narrowband as lunar illumination increases.

4.9.13 SHO Palette Campaign Scheduling

A complete Hubble-palette (SHO) image requires three narrowband channels with very different exposure needs. Typical integration ratios for a one-shot-colour (OSC) camera with dual-narrowband filter, or a mono camera cycling through dedicated filters, follow the empirical guideline Ha : Oiii : Sii $\approx 1 : 1.5 : 2$ due to the decreasing sensor QE at longer wavelengths and the typically weaker S-II emission.

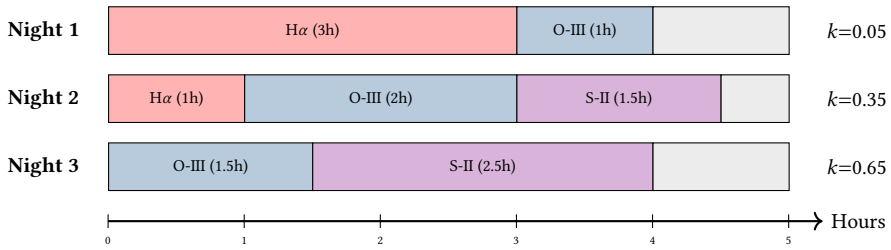


Figure 27. Multi-night SHO campaign scheduling. Under dark conditions (Night 1, $k = 0.05$), broadband-sensitive Hα is prioritised alongside initial O-III. As the Moon waxes (Nights 2–3), the schedule shifts towards O-III and S-II completion, which are more tolerant of elevated sky brightness via narrowband rejection.

The combination of the filter synergy matrix (fig. 19), the Moon-aware filter ranking (fig. 26), and the sub-frame progress tracker (fig. 25) provides the foundation for fully automated multi-night narrowband campaign management.

4.9.14 Weather, Seeing & Dew Integration

Target planning cannot be divorced from atmospheric conditions. The `safety` module provides a `WeatherReport` structure that captures temperature, humidity, dew point, wind speed, wind direction, cloud cover, rain detection, and sky quality meter (SQM) readings. These feed into the safety state machine (section 17) but also inform planning decisions:

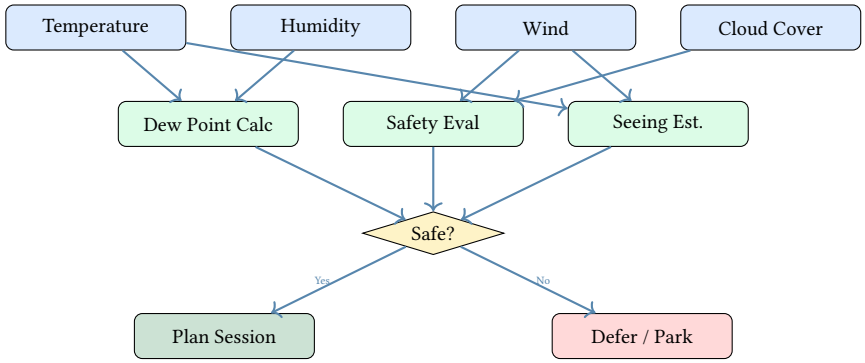


Figure 28. Weather integration in the planning pipeline. Temperature, humidity, wind, and cloud data flow through dew-point calculation, safety evaluation, and seeing estimation before the plan-or-defer decision is made.

Dew prediction. The `dew` module implements the Alduchov (1996) improved Magnus formula for dew-point calculation:

$$T_d = \frac{243.04 \cdot \gamma}{17.625 - \gamma}, \qquad \gamma = \ln(\mathrm{RH}) + \frac{17.625 \cdot T}{243.04 + T} \tag{13}$$

The *dew margin* $\Delta T_d = T_{\mathrm{OTA}} - T_d$ determines the risk of condensation on the primary optic. When $\Delta T_d < 3\,^{\circ}\mathrm{C}$, the system issues a marginal safety state and the heater duty cycle ramps linearly:

$$d = \mathrm{clamp}\left(\frac{\Delta T_{\mathrm{target}} - \Delta T_d}{\Delta T_{\mathrm{target}}},\ 0,\ 1\right) \tag{14}$$

Seeing estimation. Atmospheric seeing (characterised by the FWHM of stellar point spread functions) directly affects the achievable resolution and optimal aperture photometry radius. The `compute_snr_with_seeing` function uses the optimal aperture radius $r_{\mathrm{opt}} = 1.67 \times \mathrm{FWHM}$ (in pixels) to scale the background noise contribution. When seeing data is available from weather forecasts or real-time guide camera measurements, the planner can:

- Recommend longer focal-length targets (planetary nebulae, small galaxies) on excellent seeing nights (FWHM $< 2''$).
- Switch to wide-field mosaic targets on poor seeing nights (FWHM $> 4''$) where resolution is less critical.
- Adjust the sub-exposure time to account for the enlarged PSF's increased sky background per pixel.

4.9.15 Dark-Site Trip Planning

For urban or suburban astrophotographers (Bortle 5–8), periodic trips to dark sites (Bortle 1–3) are essential for broadband imaging of galaxies and reflection nebulae. The `site_catalog` provides a JSON interchange

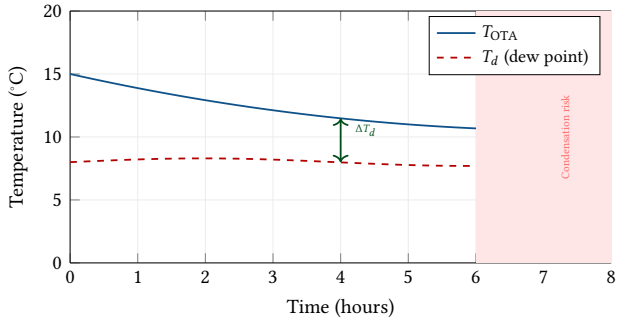


Figure 29. Dew prediction through a night. As the OTA cools radiatively, the dew margin ΔT_d narrows. The planner can use the projected crossing time to determine whether a session should be shortened or whether heater power budget will suffice.

format for multiple observing sites, each with its own location, Bortle class, elevation, tags, and optional horizon mask.

The `enrich_catalog` workflow scores the full target list against *every* catalogued site for a given date, producing an `EnrichedCatalog` that reveals the comparative advantage of each site.

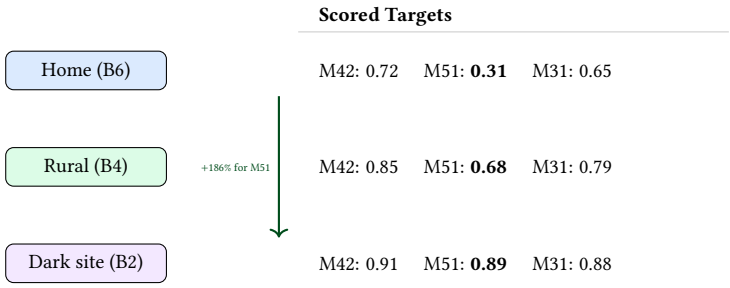


Figure 30. Dark-site comparative scoring. The Whirlpool Galaxy (M51), a broadband-sensitive face-on galaxy, benefits enormously from darker skies (+186% score improvement from B6 to B2), making it an ideal dark-site trip target. Emission-line objects like M42 show more modest gains, making them better candidates for suburban imaging with narrowband filters.

The site catalog enables a "dark-site trip optimiser" workflow:

1. Enumerate candidate dark sites within a travel radius.
2. For each site, score the full target list across the planned trip dates, weighted by lunation phase.
3. Identify the targets with the largest *score uplift*—the difference between dark-site and home-site scores—to maximise the value of the trip.
4. Cross-reference with weather forecasts and Moon phase to select the optimal trip window.

Design Decision 5: When to Travel

The optimal dark-site trip window is the intersection of three constraints: (1) new Moon ±5 days for broadband targets, (2) clear weather forecast, and (3) seasonal visibility of the desired targets. The planner's multi-factor scoring captures all three: Q_{moon} penalises bright Moon, $\bar{Q}$ integrates darkness and altitude, and f_t encodes seasonal visibility. A high composite score at a dark site during new Moon is the quantitative trigger for "book the trip."

4.9.16 Predictive Power & Mission Feasibility

Remote and portable setups run on finite battery reserves. The `power` module maintains a real-time `PowerManager` that tracks battery percentage, discharge rate (trapezoidal integration of instantaneous drain), and a configurable safe-park threshold. The planner's feasibility score p is:

$$p = \text{clamp}\left(\frac{t_{\text{safe}}}{t_{\text{block}}},\ 0,\ 1\right) \tag{15}$$

where $t_{\text{safe}} = (\text{battery\%} - \text{threshold\%})/\dot{d}$ is the estimated safe operating time remaining and t_{block} is the duration of the suggested imaging block.

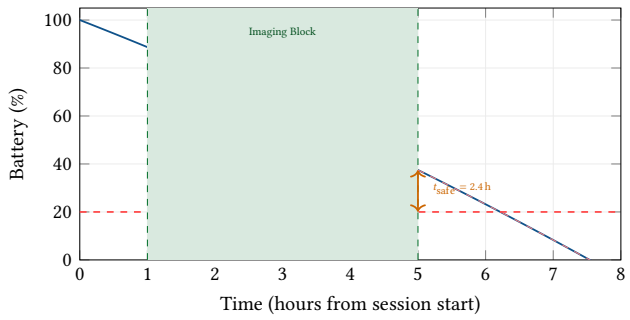


Figure 31. Predictive mission feasibility. The battery discharge curve (solid) projects forward from the current drain rate. The feasibility score p compares the remaining safe hours (orange) against the requested imaging block, allowing the planner to truncate or reject sessions that would exhaust the battery before the mount can safely park.

4.9.17 Safety State Machine Integration

The weather evaluation feeds into a priority-ordered safety state machine that can override planning decisions. The `evaluate_safety` function processes the `WeatherReport` against configurable thresholds:

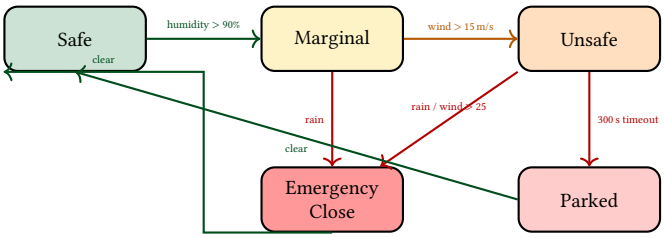


Figure 32. Safety state machine. Transitions are priority-ordered: rain and extreme wind trigger immediate emergency closure regardless of current state. Recovery to Safe requires all conditions to clear. The 300-second dwell in Unsafe before Parked prevents premature park-and-unpark cycling during intermittent gusts.

Table 1. Default safety thresholds and their planning implications.

Parameter	Default	Effect
Max wind speed	15.0 m/s	→ Unsafe
Extreme wind	25.0 m/s	→ Emergency Close
Max cloud cover	80%	→ Unsafe
Max humidity	90%	→ Marginal
Dew point margin	3.0 °C	→ Marginal
Park timeout	300 s	Unsafe → Parked

4.9.18 Horizon-Aware Visibility Curves

The `visibility` module generates sampled altitude curves via `altitude_curve`, returning altitude, azimuth, and airmass at each time step. Combined with the horizon mask, these curves provide the raw data for the planner's time-above-threshold calculation and enable rich visualisation in front-end applications.

4.9.19 IR Cloud Detection

The `ir_cloud` classifier in `astro-sentinel` provides an all-sky cloud detection pipeline from infrared sensor data. The algorithm:

1. Thresholds the IR image at a configurable warm-pixel value (default 0.35) to produce a binary cloud mask.
2. Applies 4-connected flood-fill to label contiguous warm regions.
3. Discards blobs below a minimum area (default 400 pixels) to reject stars and sensor noise.
4. Computes a detection confidence from the cloud fraction and largest blob size: $c = \text{clamp}(2f + s_{\max}/2000,\ 0.1,\ 0.99)$.

This cloud detection feeds into the safety state machine's cloud-cover input and can also inform short-term planning: if a clear patch is opening from the west, the planner can recommend targets in the clearing direction rather than abandoning the session entirely.

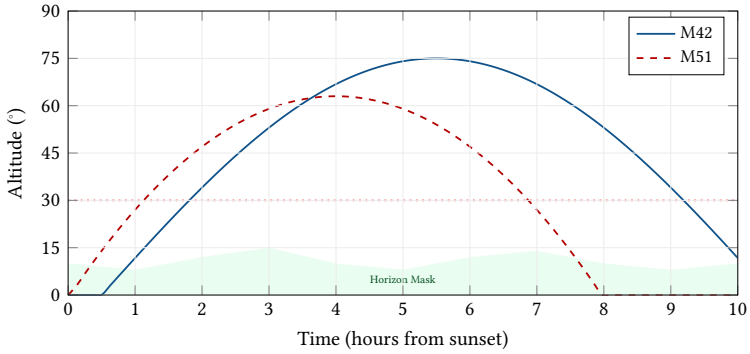


Figure 33. Horizon-aware altitude curves for two targets. The green shaded region represents the local horizon mask (trees, buildings). The effective visibility window is bounded by the maximum of the 30° minimum altitude threshold and the local horizon mask at each azimuth.

4.9.20 Unified Planning Decision Flow

Bringing all components together, the complete planning pipeline operates as follows:

The entire pipeline is composed of pure functions with no I/O or threading dependencies, enabling it to run identically on a Raspberry Pi edge daemon, a macOS desktop application, or an iOS framing assistant. The caller supplies the inputs (time, location, weather, equipment) and receives a deterministic plan—the hallmark of the SDK's pure-function architecture.

4.10 Dome Slaving Geometry

For observatories housed within a rotating hemispherical dome, the dome slit must actively track the telescope's optical axis. However, the telescope's center of rotation is rarely identical to the geometric center of the dome sphere.

The `dome` module models this as a ray-sphere intersection problem [0]. Given the mount's (X, Y, Z) spatial offset from the dome center and the dome radius R, the system projects the optical vector $\vec{v}$ (derived from the current Altitude and Azimuth) to find the exact intersection point on the dome shell, commanding the slit to the corrected azimuth.

5 Imaging Algorithms

This section presents the core imaging algorithms in `astro-vision`: noise modelling, background estimation, frame stacking, and registration.

5.1 CMOS Noise Model

The `noise_model` module encapsulates a camera's noise characteristics in a `CameraProfile` struct:

- `gain`: electrons per ADU (e^-/ADU)
- `read_noise`: RMS read noise in electrons
- `dark_current`: dark current in e^-/pixel/second
- `full_well`: full well capacity in electrons
- `bit_depth`: ADC resolution (typically 12 or 14 bits)

The SNR computation implements eq. (1) directly. When stacking N sub-exposures, the total signal scales as N while the uncorrelated noise components scale as $\sqrt{N}$, yielding a $\sqrt{N}$ improvement in SNR.

5.1.1 Optimal Sub-Exposure

The "swamp the read noise" strategy determines the minimum sub-exposure time t_{opt} at which the sky background dominates the read noise:

$$t_{\text{opt}} = \frac{(r \cdot \sigma_R)^2}{B_{\text{sky}}} \tag{16}$$

where r is a configurable ratio factor (default 3.0) and B_{sky} is the sky background rate in e^-/pixel/s. At this exposure time, the sky background noise is r times the read noise, ensuring that read noise contributes less than $1/(r^2 + 1) \approx 10\%$ of the total noise variance.

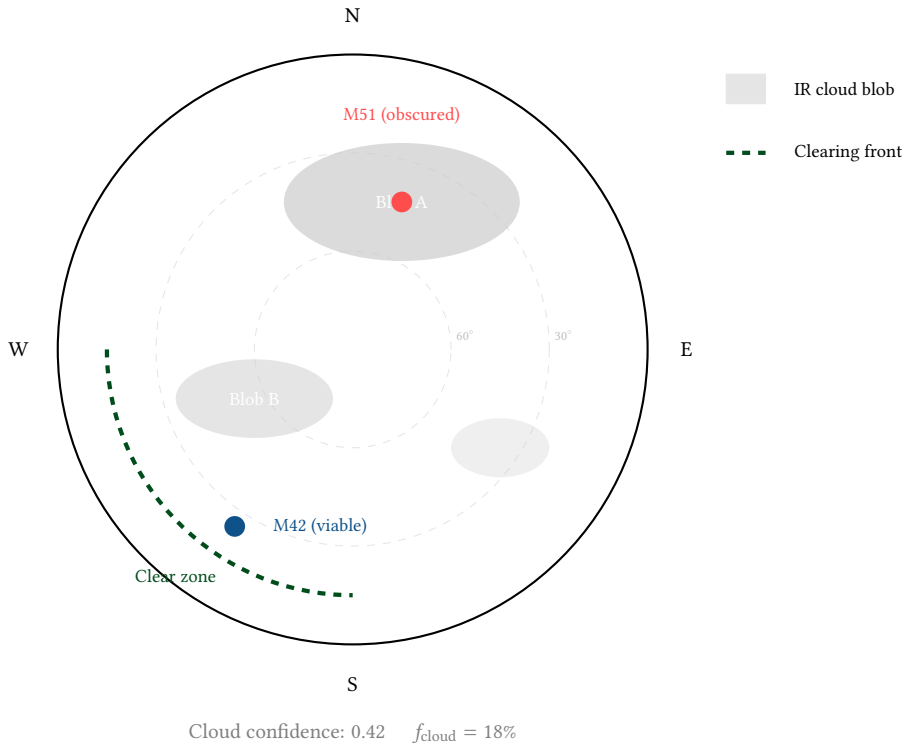


Figure 34. IR all-sky cloud map. Cloud blobs (grey) are identified via flood-fill on the thresholded IR image. Targets in clear zones (M42) remain viable; targets obscured by clouds (M51) are deprioritised by the real-time planner. The dashed arc indicates an approaching clearing front from the south-west.

> **Design Decision 6: Swamp Ratio Default**
>
> The default ratio $r = 3.0$ ensures read noise is <10% of total variance. Higher ratios (e.g. $r = 5$) reduce the read noise contribution further but require proportionally longer sub-exposures, which may exceed the saturation limit at bright sites. The value 3.0 represents the community consensus for a practical compromise.

5.1.2 Sky Brightness Model

Sky background is derived from the Bortle scale via a magnitude-to-flux conversion. The nine-level `BortleScale` enum maps each class to a sky surface brightness in mag/arcsec^2 (ranging from 22.0 at Bortle 1 to 17.5 at Bortle 9):

Bortle Class	SB (mag/arcsec^2)	Description
B1	22.0	Excellent dark site
B2	21.8	Typical dark site
B3	21.6	Rural sky
B4	21.0	Rural/suburban transition
B5	20.0	Suburban sky
B6	19.0	Bright suburban
B7	18.5	Suburban/urban transition
B8	18.0	City sky
B9	17.5	Inner-city sky

The flux conversion accounts for the telescope aperture area, pixel scale, and sensor quantum efficiency. Built-in camera profiles are provided for the ZWO ASI294MC Pro, ASI533MC Pro, ASI2600MC Pro, and Canon 6D (modified).

5.1.3 Spectral Response & Narrowband Physics

The `CameraProfile` also characterizes the sensor's spectral Quantum Efficiency (QE), crucial for narrowband imaging (e.g., Hydrogen-alpha, Oxygen-III, Sulfur-II).

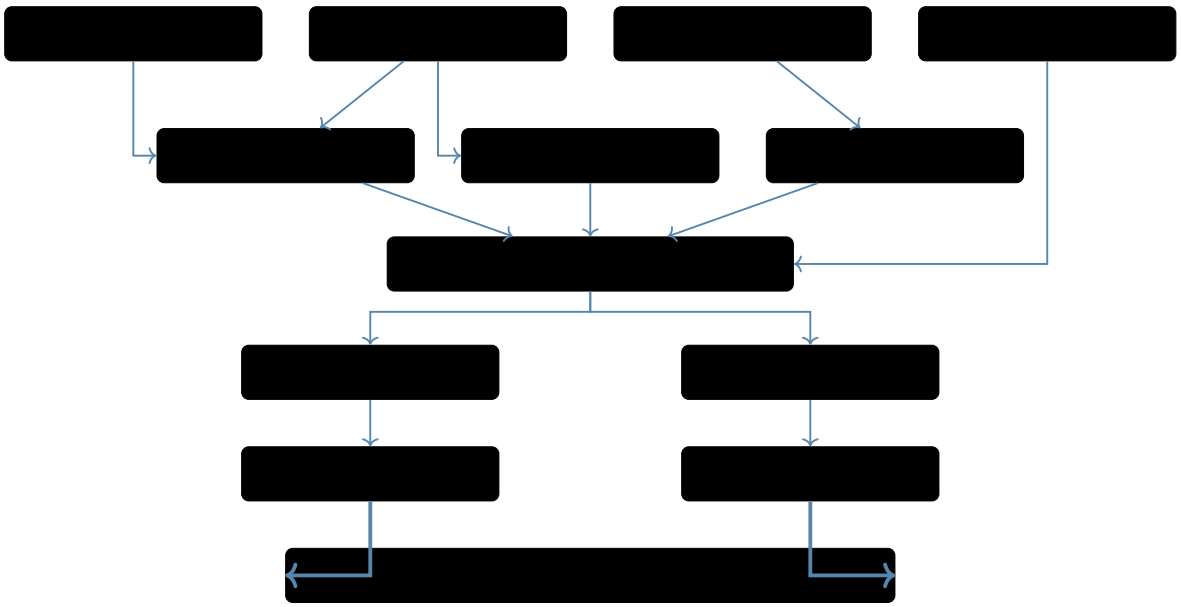


Figure 35. Unified planning decision flow. Equipment configuration, site catalog, target list, and weather data feed through twilight computation, Moon ephemeris, and visibility analysis into the scoring engine. The output is a concrete, actionable recommendation: which target, when, with which filter, how many subs, and the predicted SNR.

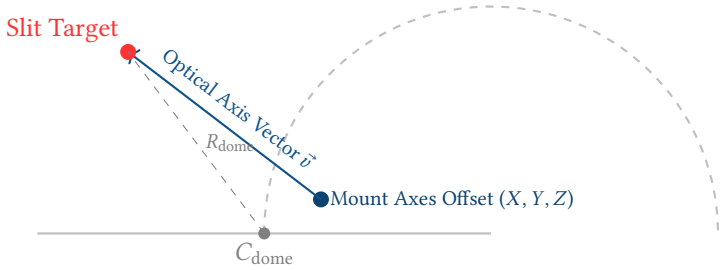


Figure 36. Dome slaving kinematics. Because the telescope mount is offset from the dome's geometric center, the optical axis $\vec{v}$ does not point to the same azimuth as the mount's pointing azimuth. The system must solve the ray-sphere intersection to command the correct dome rotation angle.

By incorporating the QE curve, the pipeline can accurately predict exposure requirements for narrowband targets, accounting for the reduced sensor sensitivity in the deep red (H-α, S-II) compared to the peak sensitivity in the green (O-III).

5.2 Bayer CFA Demosaicing

Raw sensor data from color cameras is captured through a Color Filter Array (CFA), typically in an RGGB Bayer pattern. The `palette` module reconstructs the full RGB image using high-quality linear interpolation [0].

The algorithm calculates cross-channel gradients to detect edges, dynamically weighting the interpolation to prevent cross-color artifacts (zippering) along sharp stellar boundaries.

5.3 Background Estimation

Accurate background subtraction is prerequisite to star detection and photometry. The `background` module estimates a spatially varying background model through a four-step process:

1. **Tiling.** The image is divided into a grid of 64×64 pixel tiles.
2. **Per-tile statistics.** A sigma-clipped mean is computed for each tile (3σ, 5 iterations), yielding a robust estimate of the local background level.
3. **Tile rejection.** Tiles whose mean deviates from the cross-tile median by more than 2σ are rejected. These tiles typically contain bright stars, nebulosity, or sensor defects.
4. **Interpolation.** The surviving tile values are bilinearly interpolated [0] to produce a full-resolution background map.

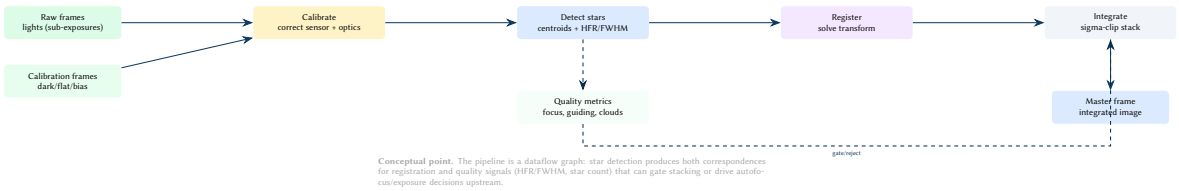


Figure 37. Imaging as a dataflow graph with control signals. Solid arrows carry pixel data; dashed arrows carry derived metrics that inform decisions.

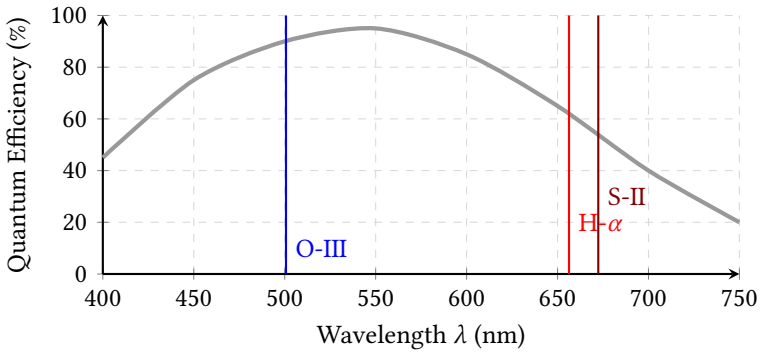


Figure 38. Quantum Efficiency curve of a typical Back-Illuminated (BSI) CMOS sensor. The noise model calculates the expected signal-to-noise ratio by integrating the target's emission spectrum against the sensor's specific QE at key narrowband wavelengths (500.7 nm, 656.3 nm, 672.4 nm).

> **Tile Size Choice**
>
> The 64×64 tile size balances spatial resolution against statistical robustness. Smaller tiles resolve sharper gradients but contain fewer pixels, making sigma-clipping less reliable. For most astronomical images ($\geq$1024×1024 pixels), 64-pixel tiles provide sufficient gradient resolution while ensuring each tile contains at least 4,096 samples.

5.4 Star Detection

Star detection follows a threshold-and-flood-fill approach:

1. **Threshold.** Pixels above $\mu_{\text{bg}} + k \cdot \sigma_{\text{bg}}$ are marked as candidates (default $k = 5$).
2. **Connected components.** 4-connectivity flood fill labels contiguous candidate regions.
3. **Filtering.** Regions outside the area bounds $[A_{\min}, A_{\max}]$ are rejected.
4. **Centroiding.** The intensity-weighted centre of mass is computed for each surviving region.
5. **HFR.** Pixel distances from the centroid are sorted and accumulated until 50% of the total flux is enclosed; this radius is the HFR. The FWHM is estimated as $\text{FWHM} = 2 \cdot \text{HFR} \cdot \sqrt{\ln 2}$.

For advanced star extraction, particularly when tracking faint guide stars against variable moonlit backgrounds, the pipeline supplements the flood-fill approach with a scale-space Laplacian of Gaussian (LoG) blob detector [0]. The filter's zero-crossings uniquely identify the inflection boundaries of the stellar profile regardless of the local DC background level.

Insight: *The HFR metric is preferred over FWHM for autofocus because it is model-free: it does not assume a Gaussian PSF, making it robust to optical aberrations, coma, and field curvature.*

5.5 Frame Stacking

The `stacking` module combines registered sub-exposures pixel-by-pixel using one of four methods.

5.5.1 Mean and Median

The arithmetic mean maximises SNR under Gaussian noise but is sensitive to outliers (satellite trails, cosmic rays). The pixel-wise median provides robust outlier rejection but sacrifices ~20% of the theoretical SNR compared to the mean.

5.5.2 Sigma-Clipped Mean

The default stacking method iteratively rejects outlier pixels:

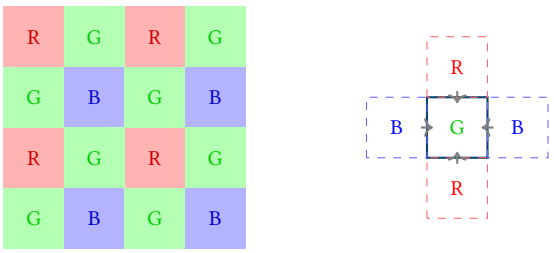


Figure 39. Demosaicing process. The missing color channels at any given pixel are computed using gradient-corrected linear interpolation from neighboring pixels to preserve sharp edges and minimize false-color artifacts.

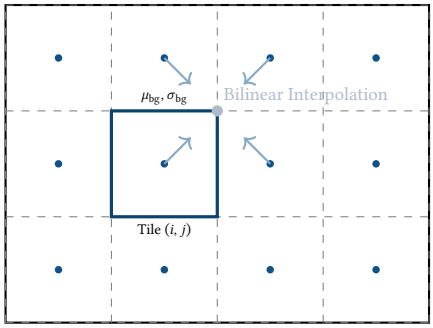


Figure 40. Background estimation using a local grid. Each tile computes a sigma-clipped mean, which is then bilinearly interpolated across the image to model spatial gradients.

Algorithm 1 Sigma-clipped mean stacking (per pixel).

Require: Pixel values $\{p_1, \ldots, p_N\}$, rejection threshold κ, max iterations I
Ensure: Stacked value $\bar{p}$

$S \leftarrow \{p_1, \ldots, p_N\}$
for $i = 1$ to I **do**
  $\mu \leftarrow \text{mean}(S)$; $\sigma \leftarrow \text{std}(S)$
  $S' \leftarrow \{p \in S : |p - \mu| \leq \kappa\sigma\}$
  if $|S'| = |S|$ **then break** ▷ converged
  end if
  $S \leftarrow S'$
end for
return $\text{mean}(S)$

The default parameters ($\kappa = 2.5$, $I = 5$) reject approximately 1.2% of values per iteration under a Gaussian distribution. If all pixels are clipped (degenerate case), the method falls back to the full-sample mean.

Four stacking methods are supported, each trading robustness against computational cost:

Method	Behavior	Default
Mean	Simple arithmetic mean; fastest but vulnerable to outliers	—
Median	Order-statistic; robust to $< 50\%$ outliers but higher variance than clipped mean	—
Sigma-Clipped Mean	Iteratively rejects pixels $> \kappa\sigma$ from the mean, then averages survivors	$\kappa = 2.5$, 5 iter
Winsorized Sigma Clip	Replaces outliers with boundary values rather than discarding; preserves sample size	$\kappa = 2.5$, 5 iter

The default method is sigma-clipped mean with $\kappa = 2.5$ and a maximum of 5 iterations, which provides the best balance of outlier rejection (satellite trails, cosmic rays) and noise reduction for typical amateur deep-sky sessions.

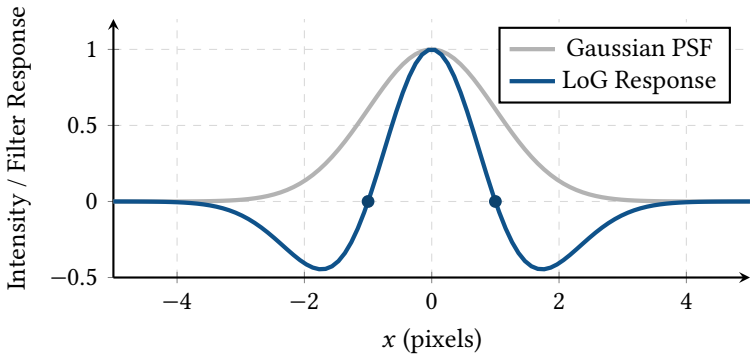


Figure 41. 1D profile of a Gaussian star (grey) and the normalized Laplacian of Gaussian (LoG) filter response (blue). The zero-crossings of the LoG function formally define the inflection edges of the star's spatial profile, allowing for robust multi-scale star detection invariant to background gradients.

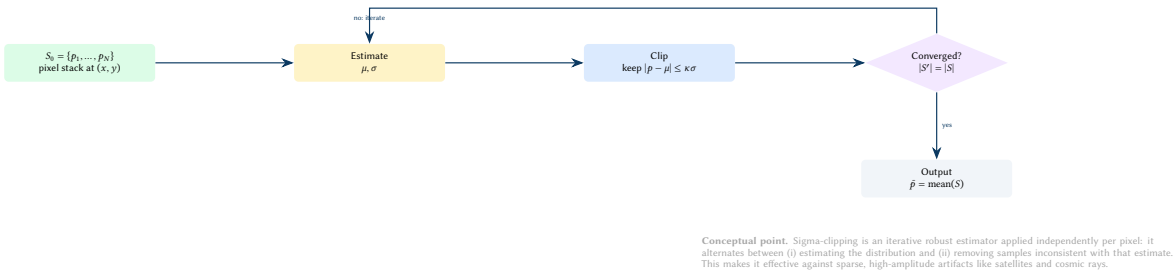


Figure 42. Sigma-clipped stacking as an iterative estimator. The same loop runs per pixel across the registered frame stack.

5.5.3 Winsorized Sigma Clipping

The Winsorized variant [0]—the stacking method used by PixInsight's default pipeline—replaces outlier pixels with the boundary values rather than discarding them:

$$p_i' = \begin{cases} \mu - \kappa\sigma & \text{if } p_i < \mu - \kappa\sigma \\ \mu + \kappa\sigma & \text{if } p_i > \mu + \kappa\sigma \\ p_i & \text{otherwise} \end{cases} \tag{17}$$

The Winsorized values are then averaged. This preserves the sample size N and therefore the $\sqrt{N}$ SNR improvement, while still suppressing outlier influence.

> **Design Decision 7: Winsorized vs. Sigma-Clipped Stacking**
>
> Sigma clipping discards outliers entirely, reducing the effective number of samples and introducing a small bias towards the centre of the distribution. Winsorizing retains all samples by replacing extremes with boundary values, achieving similar outlier rejection with marginally better SNR. The difference is most visible in stacks of fewer than ~20 frames, where each discarded pixel represents a significant fraction of the sample.

5.6 Exposure Intelligence

The `exposure` module synthesises the noise model, sky brightness model, and camera profile into actionable exposure recommendations:

1. Compute t_{opt} from the "swamp the read noise" formula (eq. (16)) at $r = 3.0$.
2. Clamp to 80% of the saturation time to prevent clipping: $t_{\max} = 0.8 \cdot C_{\text{FW}}/(S + B_{\text{sky}} + D)$.
3. Clamp to the total integration budget: $t \le T_{\text{total}}$.
4. Enforce a minimum of 1 s to avoid shutter-timing artefacts.
5. Derive the number of sub-exposures $n = T_{\text{total}}/t$ and the predicted SNR at the total integration.

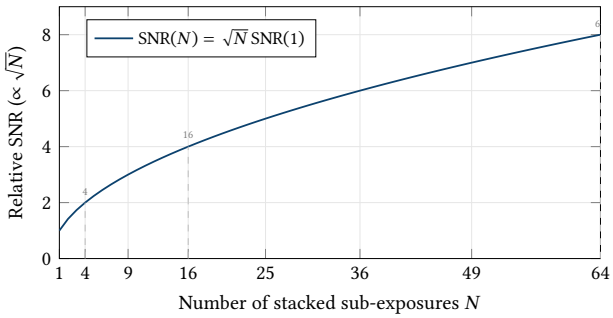


Figure 43. SNR improvement from stacking N registered sub-exposures. Because uncorrelated noise components average down as $1/\sqrt{N}$, each doubling of frame count yields a $\sqrt{2} \approx 1.41\times$ improvement. The curve illustrates the law of diminishing returns: the jump from 1 to 4 frames is dramatic (2× SNR), while going from 16 to 64 frames only doubles it again.

5.6.1 Optimal Sub-Exposure Time

A foundational principle of modern CMOS astrophotography is that once the background sky noise exceeds the camera's read noise by a sufficient margin (typically 10×), longer sub-exposures provide no additional signal-to-noise benefit to the final stacked image [0].

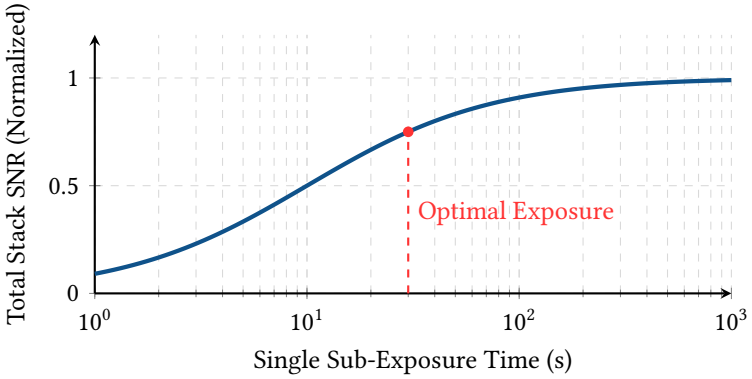


Figure 44. Diminishing returns of sub-exposure time. Once the t_{opt} threshold is reached (where sky noise swamps read noise), extending the exposure time only increases the risk of ruined frames (e.g., from satellite trails or wind gusts) without improving the final stack's SNR.

The `exposure` module analytically solves this threshold based on the user's Bortle sky class and the sensor's read noise profile, automatically splitting a requested 2-hour integration into the optimal number of mathematically equivalent short sub-exposures.

5.6.2 Holy Grail Timelapse Ramp

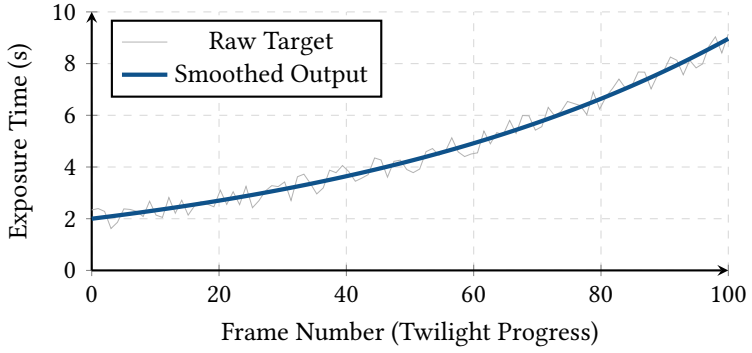


Figure 45. Holy Grail timelapse exposure ramping. The exponential smoothing filter rejects high-frequency noise from passing clouds or atmospheric variations, generating a monotonically increasing exposure schedule that prevents visible flicker.

For day-to-night transition timelapses, the `holy_grail_ramp` function computes a smooth exposure ramp. At each frame i, the raw exposure is $t_i = E_{\text{target}}/B_{\text{sky}}(i)$, where $B_{\text{sky}}(i)$ decreases as twilight deepens. A 20%-per-step exponential smoothing filter prevents visible flicker between adjacent frames.

5.7 Frame Registration

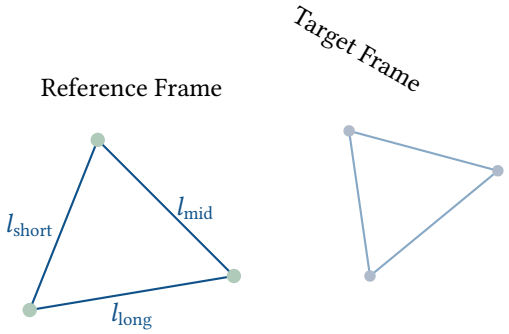


Figure 46. Triangle-similarity matching. The sorted ratios of the sides ($l_{\text{short}}/l_{\text{long}}$, $l_{\text{mid}}/l_{\text{long}}$) form a scale-, translation-, and rotation-invariant descriptor.

The `registration` module aligns sub-exposures to a reference frame using triangle-similarity matching [0]:

1. **Star selection.** The N brightest stars are selected from each frame (default $N = 30$).
2. **Triangle formation.** All $\binom{N}{3}$ triangles are formed from the selected stars.
3. **Descriptor computation.** Each triangle is described by the sorted ratio pair ($\ell_{\text{short}}/\ell_{\text{long}}$, $\ell_{\text{mid}}/\ell_{\text{long}}$), which is invariant to translation, rotation, and uniform scaling.
4. **Matching.** Triangles from the reference and target frames are matched by Euclidean distance in descriptor space.
5. **RANSAC affine fit.** Matched triangle pairs provide star correspondences. A RANSAC-style loop [0] selects three correspondences, solves for the 6-parameter affine transform via Cramer's rule, counts inliers within a 5-pixel tolerance, and retains the best model.
6. **Reprojection.** The target frame is resampled onto the reference grid using inverse-transform bilinear interpolation.

> **Affine vs. Projective**
>
> An affine transform (6 parameters) is sufficient for astronomical images because the field of view is small enough that perspective distortion is negligible. Projective transforms (8 parameters) would require a minimum of 4 correspondences per RANSAC trial and offer no practical benefit for the $< 5^\circ$ fields typical of deep-sky imaging.

The affine transform $\mathbf{A}$ maps points (x, y) from the target to the reference:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{18}$$

Given three point correspondences, the system is exactly determined and solved by Cramer's rule in $O(1)$. The inverse transform (used for reprojection) requires the determinant $ad - bc \neq 0$.

5.8 Calibration Pipeline

The `calibrate` module implements the standard astrophotography calibration formula:

$$I_{\text{corrected}} = \frac{I_{\text{raw}} - I_{\text{dark}}}{I_{\text{flat}} - I_{\text{bias}}} \tag{19}$$

where the flat field is normalised by its mean. The module also provides:

- **Hot pixel detection** via MAD-based sigma estimation ($\hat{\sigma} \approx 1.4826 \cdot \text{MAD}$) with thresholding above median + $\kappa\hat{\sigma}$.
- **Cold pixel detection** as pixels below a fraction of the flat frame median.
- **Bad pixel interpolation** via mean of valid 3×3 neighbours.

5.9 Spatial Dithering & Fixed-Pattern Noise

Even with ideal calibration, modern sensors exhibit Fixed-Pattern Noise (FPN) and localized defects that remain static relative to the pixel grid. If the mount tracks perfectly, this noise accumulates coherently during stacking, resulting in correlated "walking noise."

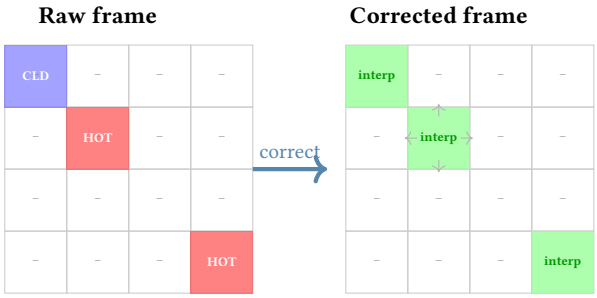


Figure 47. Hot and cold pixel correction in the `calibrate` module. Defective pixels (red: hot, blue: cold) are identified via MAD-thresholding on the dark master. Each flagged pixel is replaced by the mean of its valid 3×3 neighbours (green), removing permanent sensor defects before registration and stacking.

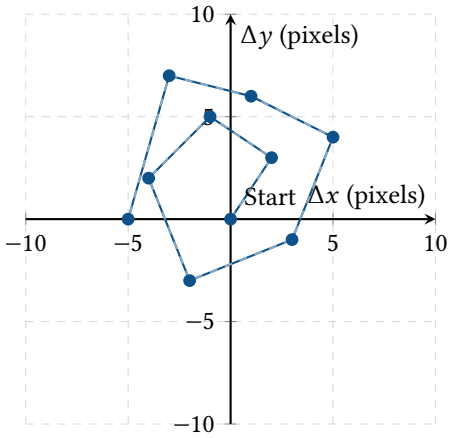


Figure 48. A pseudo-random Dither map over 10 exposures. Between each frame, the mount is commanded to shift slightly. While the stars are re-aligned during registration, the fixed-pattern sensor noise is decoupled and scattered across the grid, allowing the sigma-clipper to successfully reject it.

The `sequencer` integrates spatial dithering (sometimes paired with Drizzle integration [0]) by randomly shifting the mount's aim point by a few arcseconds between frames. During registration, the stars are translated back to a common origin, which mathematically scatters the FPN across the spatial domain, allowing the Winsorized Sigma Clipper to easily reject the defects as stochastic outliers.

5.10 Non-Linear Image Stretching

Raw stacked astronomical images are highly linear and appear nearly pitch black to the human eye, as the faint nebulosity data is compressed into the lowest histogram bins. The `stretch` module applies a Midtone Transfer Function (MTF) [0] to execute a non-linear histogram stretch.

The stretch algorithm maps the input interval defined by the Black Point (BP) and White Point (WP) to $[0, 1]$ and solves the rational MTF equation to shift the image's median brightness to a target midtone value (typically 0.25 for aesthetic rendering).

5.11 Strict FITS Metadata Provenance

Scientific image processing requires rigorous data provenance. A stacked Master image is mathematically useless if its temporal origins, physical exposure duration, or calibration history are lost. The SDK ensures strict metadata preservation as frames flow through the typestate pipeline.

The `fits_write` module actively manages keyword inheritance. It strips camera-specific telemetry (like USB bandwidth flags) that are irrelevant post-calibration, computes the exact 'DATE-OBS' midpoint for time-domain astronomy, and injects WCS coordinate matrices when spatial transformations occur.

6 World Coordinate System

The `wcs` module in `astro-core` implements the FITS WCS [0] with the gnomonic (TAN) projection and Simple Imaging Polynomial (SIP) distortion corrections [0].

6.1 TAN Projection

The TAN projection maps pixel coordinates (u, v) to intermediate world coordinates (ξ, η) on the tangent plane via the CD matrix:

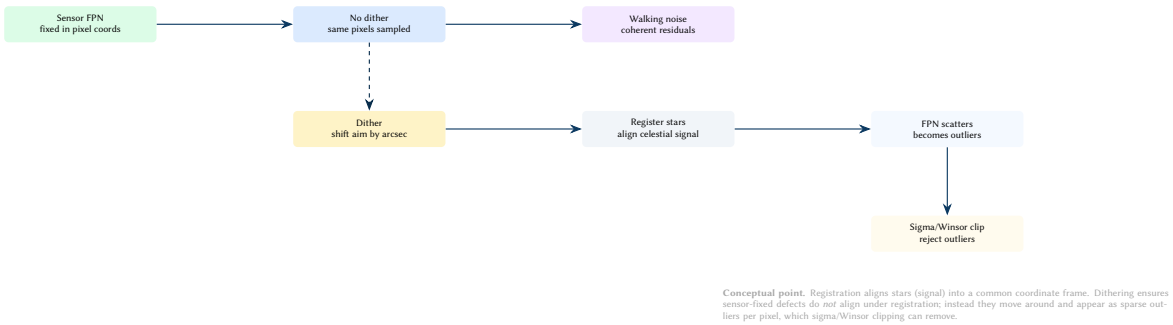


Figure 49. Why dithering works: it breaks coherence of sensor-fixed noise under registration, converting correlated defects into rejectable outliers in the stacked frame.

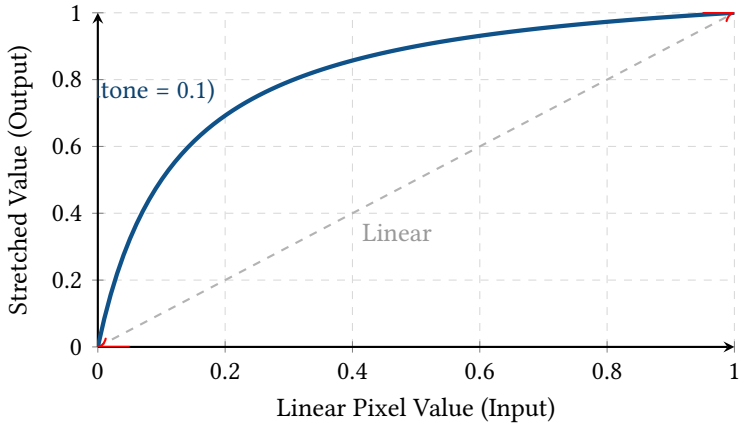


Figure 50. The Midtone Transfer Function (MTF) aggressively expands the dynamic range of the dark background (pulling faint signal up) while compressing the highlights (preventing star cores from blowing out).

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \begin{pmatrix} CD_{1,1} & CD_{1,2} \\ CD_{2,1} & CD_{2,2} \end{pmatrix} \begin{pmatrix} u - CRPIX_1 \\ v - CRPIX_2 \end{pmatrix} \tag{20}$$

The de-projection from (ξ, η) to sky coordinates (α, δ) uses the standard gnomonic inverse:

$$\delta = \arcsin\left(\frac{\sin\delta_0 + \eta \cos\delta_0}{\sqrt{1 + \xi^2 + \eta^2}} \right) \tag{21}$$

$$\alpha = \alpha_0 + \arctan\left(\frac{\xi}{\cos\delta_0 - \eta \sin\delta_0} \right) \tag{22}$$

6.2 SIP Distortion

Real optics introduce radial and tangential distortion that the linear CD matrix cannot represent. The SIP convention adds forward polynomial corrections $A(u, v)$ and $B(u, v)$ to the pixel offsets before the CD matrix is applied:

$$u' = u + \sum_{p+q \le N} A_{p,q}\, u^p\, v^q \tag{23}$$

$$v' = v + \sum_{p+q \le N} B_{p,q}\, u^p\, v^q \tag{24}$$

where N is the polynomial order (typically 3–5). The inverse transform uses the published AP/BP polynomials if available; otherwise, a fixed-point iteration converges to the undistorted pixel coordinates:

$$u_{k+1} = u_{\text{target}} - A(u_k, v_k), \quad v_{k+1} = v_{\text{target}} - B(u_k, v_k) \tag{25}$$

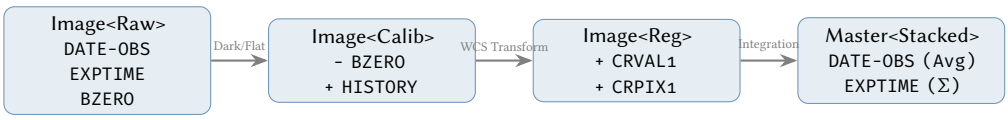


Figure 51. FITS Metadata Provenance. As an image traverses the pipeline, its header keywords are mutated to reflect the mathematical transformations. For example, during stacking, the `EXPTIME` is summed, while the `DATE-OBS` is averaged to the exact mid-point of the total integrated exposure span.

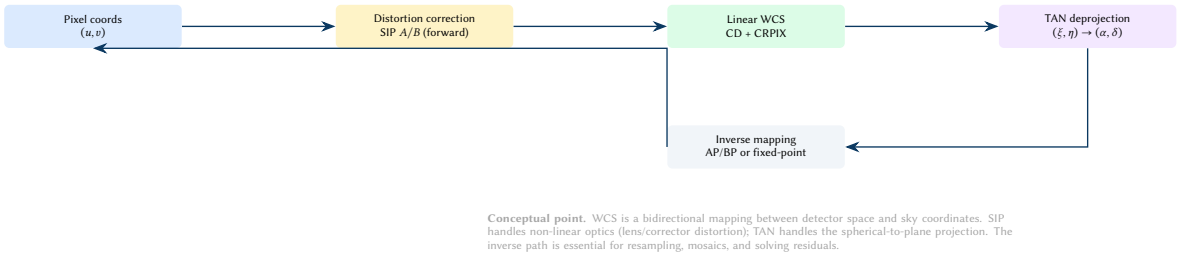


Figure 52. WCS/SIP as a bidirectional mapping. Forward maps pixels to sky coordinates; inverse maps sky to pixels for resampling and mosaic planning.

The iteration converges in 3–5 steps for typical distortion magnitudes (<10 pixels at the field edge) with a tolerance of 10^{-12} pixels.

> **Design Decision 8: Fixed-Point vs. Newton Inversion**
>
> Newton's method would converge in fewer iterations but requires computing the Jacobian of the SIP polynomials at each step. Since the distortion is small relative to the pixel scale, the fixed-point iteration converges rapidly and avoids the implementation complexity and numerical risk of Jacobian inversion. The maximum iteration count (30) provides a safety bound that is never reached in practice.

6.3 Derived Quantities

The WCS module provides several derived quantities used by the framing assistant and mosaic planner:

- `pixel_scale_arcsec_per_px`: extracted from the CD matrix determinant.
- `image_fov_deg`: field of view in RA and Dec from the pixel scale and image dimensions.
- `framing_offset_to`: east/north arcsecond offset needed to recentre a target, enabling "nudge" commands in the iOS app.

7 Design Rationale

This section discusses the design decisions that shape the SDK's architecture.

7.1 Compile-Time Pipeline Safety (Typestate Pattern)

Astrophotography mandates a strict sequence of operations: raw frames must be calibrated, then star-detected, then registered, before they can be stacked. Attempting to stack unregistered frames yields a blurred failure.

The `astro-vision` crate encodes this state machine directly into the type system using the Typestate Pattern [0].

By using zero-cost marker types (`Raw`, `Calibrated`, `Registered`), pipeline logic errors become compile-time errors rather than runtime panics or silent data corruption.

7.2 Algorithmic Complexity & SIMD Vectorization

Processing modern 60-megapixel (60M) sensor arrays requires extreme computational efficiency. The SDK relies heavily on the `ndarray` crate, which transparently compiles vector and matrix operations down to AVX-512 and NEON SIMD instructions.

For geometric algorithms like triangle-matching registration, the naive formulation requires $O(N^3)$ complexity to form all possible triangles from N stars. The pipeline bounds this to $O(N \log N)$ by pre-filtering stars by SNR and indexing the reference descriptors in a k-d tree.

7.3 Golden Master Ephemeris Validation

Mathematical purity is only useful if the underlying physics model is empirically correct. The `astro-core` crate verifies its positional astronomy pipeline against established scientific baselines through continuous automated testing.

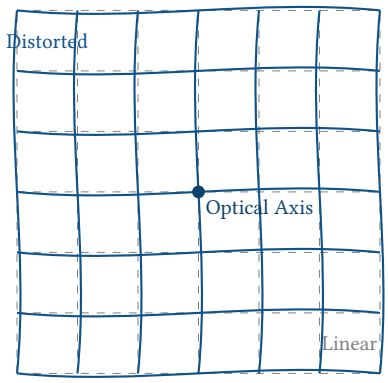


Figure 53. Visualization of SIP polynomial distortion warping the linear TAN projection grid away from the optical axis.

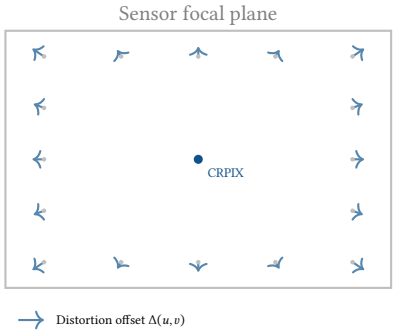


Figure 54. SIP distortion field for a representative wide-field lens exhibiting barrel distortion. Each arrow shows the $(\Delta u, \Delta v)$ pixel correction applied by the A/B polynomials before the linear CD matrix maps to sky coordinates. The correction magnitude is greatest at the corners and falls to zero at the optical centre (CRPIX).

The test suite compares generated topocentric coordinates, eclipse contact times, and nutation matrices against the JPL Horizons On-Line Ephemeris System [0]. A test passes only if the residuals fall below the defined sub-arcsecond tolerance threshold, proving the fidelity of the SDK's internal float representations.

7.4 The Pure-Function Constraint

: Functional Purity

Every public function in `astro-core` and `astro-vision` must be a pure function: given the same inputs, it must produce the same output, with no side effects. Neither crate may depend on `tokio`, `std::fs`, or any I/O library.

The benefits are threefold:

1. **Testing.** Unit tests are simple assertions with no mocks, fixtures, or cleanup. The 275 tests across the workspace run in under 2 seconds.
2. **FFI.** Pure functions with scalar or struct arguments are trivially exposed via C ABI for consumption by Swift, Python, or JavaScript.
3. **Determinism.** Identical inputs produce identical outputs on any platform, simplifying cross-platform debugging and enabling golden-file testing.

7.5 Newtype Discipline

: Physical Quantity Newtypes

All physical quantities must be wrapped in newtypes. A function expecting `Latitude` cannot receive a `Declination`, even though both are measured in degrees. Direct use of raw `f64` values for angular quantities is prohibited in the public API.

Figure 55. The Typestate pattern enforces the astrophotography pipeline at compile-time. The `stack()` method is only implemented for the `Image<Registered>` state, making it mathematically impossible to execute pipeline stages out of order.

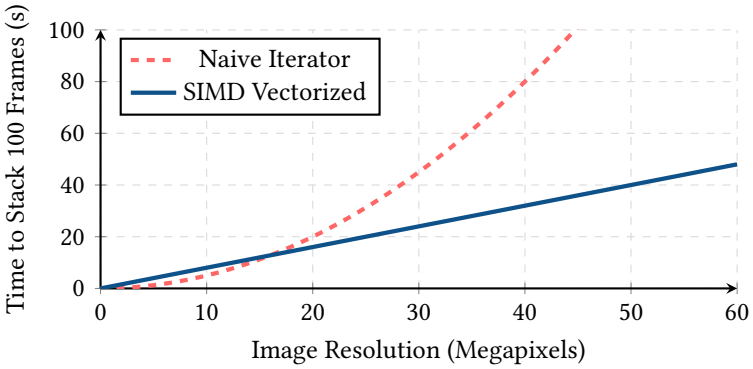


Figure 56. Theoretical scaling performance of Sigma-Clipped stacking. By relying on contiguous memory layouts and SIMD intrinsic vectorization, the `stacking` module maintains a linear $O(N)$ time complexity, preventing the combinatorial explosion associated with naive per-pixel loop evaluation on large sensors.

A Lesson from Mars

The Mars Climate Orbiter was lost because one software module produced thrust in pound-force·seconds while the consuming module expected newton·seconds [0]. In astrophotography, the analogous risk is confusing RA (hours) with longitude (degrees) or altitude (above horizon) with declination (above equator).

7.6 Error Handling

The SDK enforces a strict "no panic" discipline: every fallible operation returns a typed `Result<T, E>` with a domain-specific error enum rather than panicking or returning sentinel values. Each module defines its own error type (e.g. `StackError`, `WcsError`, `CalibrationError`) with descriptive variants carrying the context needed to diagnose the failure.

This layered approach provides three key guarantees:

1. **No silent failures**: A failed plate solve cannot be mistaken for a successful one—the caller must explicitly handle the error variant or propagate it upward.
2. **Contextual diagnostics**: Error variants carry the originating parameters (e.g., the number of stars found vs. required, the timeout duration exceeded), enabling automated retry logic to adjust strategy rather than blindly retrying.
3. **FFI safety**: At the C ABI boundary, typed errors are mapped to integer codes with a companion `last_error_message` function that returns a human-readable description, preserving diagnostic information across the language boundary.

7.7 ndarray as the Image Primitive

Design Decision 9: Why ndarray, Not a Custom Image Type?

Image data flows through the SDK as `Array2<f32>` from `ndarray` rather than a custom image struct. This trades a small amount of type safety (e.g. no compile-time enforcement of the $[0, 1]$ normalisation invariant) for interoperability: any library that operates on `ndarray` arrays can be composed with the SDK without conversion overhead.

The thin `AstroImage` wrapper adds metadata (bit depth, exposure time) without restricting access to the underlying array.

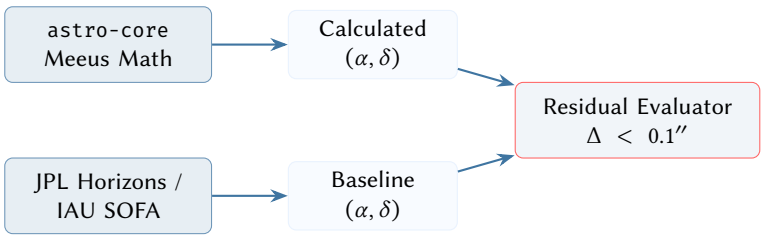


Figure 57. Golden Master CI/CD testing pipeline. The SDK's coordinate output is dynamically asserted against the gold-standard JPL Horizons ephemeris data [0] to guarantee that the implementation is free from systemic truncations or coordinate frame misinterpretations.

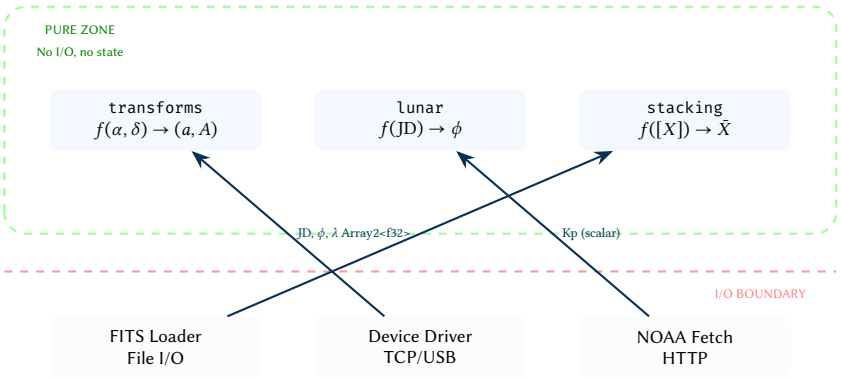


Figure 58. The pure-function boundary. Foundation crates (`astro-core`, `astro-vision`) exist entirely within the pure zone. External data crosses the I/O boundary as scalar or array parameters—never via side effects.

8 Autofocus & Collimation

8.1 V-Curve Autofocus

The `autofocus` module fits a V-curve model to HFR measurements across focuser positions:

$$\mathrm{HFR}(x) = b + m \cdot |x - c| \tag{26}$$

where c is the optimal focus position, b is the minimum HFR, and m is the defocus slope.

The fitting procedure uses **IRLS** (Iteratively Reweighted Least Squares) with Huber weights [0] ($\psi(r) = \min(|r|, k) \cdot \mathrm{sign}(r)$, $k = 1.345$) for robustness to outlier frames (e.g. a frame with a passing satellite inflating the HFR).

Insight: *The V-curve model assumes that defocus produces a linear increase in HFR—a valid approximation for small defocus excursions typical of automated focusing runs.*

The module enforces *informativeness gates* that reject non-diagnostic focus runs:

- **Flat scan**: insufficient HFR variation across the range, suggesting the focuser did not move enough.
- **Unbracketed**: the minimum HFR is at one end of the scan, suggesting the best focus lies outside the sampled range.
- **Edge minimum**: similar to unbracketed, with a directional suggestion for the next scan.

8.2 Thermal Expansion & Focus Compensation

As the ambient temperature drops throughout the night, the metal chassis of the telescope contracts, shifting the focal plane inward. Running a full V-Curve autofocus routine takes several minutes and interrupts data acquisition.

The thermal compensation model is:

$$\Delta f = \alpha \cdot f_L \cdot \Delta T, \qquad \Delta_{\mathrm{steps}} = \mathrm{round}\left(\Delta f \cdot n_{\mathrm{steps/mm}}\right) \tag{27}$$

where α is the coefficient of thermal expansion (CTE) of the telescope's structural material, f_L is the focal length, and ΔT is the temperature change. The SDK provides built-in CTE values:

8.3 IRLS Fitting Details

The V-curve fitting procedure (Eq. 26) uses IRLS with Huber weighting to achieve breakdown-point robustness. The iterative procedure (up to 12 iterations) operates as follows:

Insight: *Carbon fiber tubes require approximately 46× less focus compensation than aluminum refractors of equal focal length. For an $f_L = 1000$ mm aluminum scope and $\Delta T = -5^{\circ}$C, the shift is -0.116 mm ($\sim$ 12 steps at 100 steps/mm), while the CFRP equivalent shifts only -0.0025 mm.*

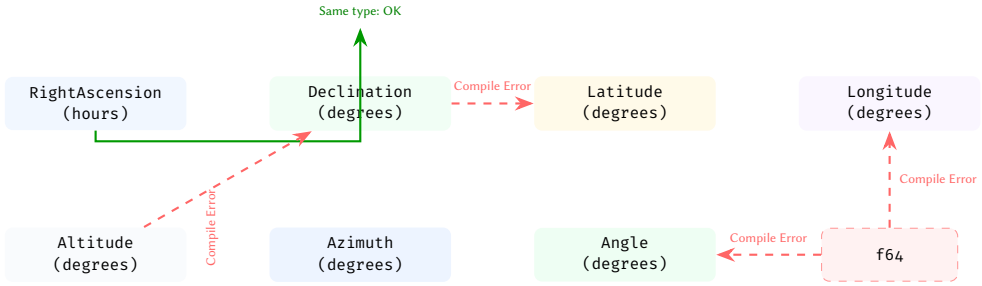


Figure 59. Newtype discipline prevents unit confusion at compile time. Although `Latitude` and `Declination` are both measured in degrees, they are distinct types and cannot be interchanged. Raw `f64` values cannot substitute for any angular newtype.

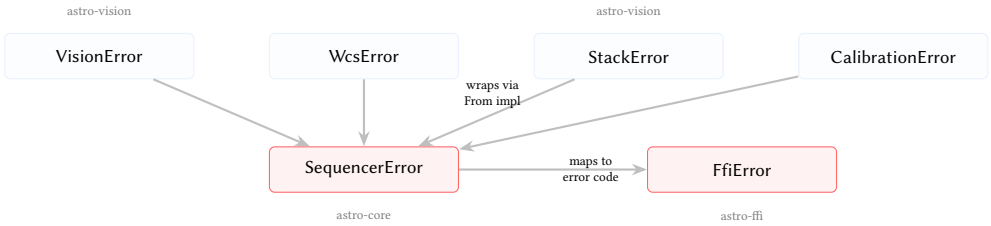


Figure 60. Error propagation hierarchy. Module-level errors are composable via `From` trait implementations, allowing low-level vision errors to propagate through the sequencer to the FFI boundary, where they are mapped to integer error codes for C callers.

1. **Grid Search**: 64 candidate c positions (including all sample positions and midpoints) are evaluated. For each candidate, the transformed variable $t_i = |x_i - c|$ reduces the V-curve to an ordinary linear model $y = b + m \cdot t$.
2. **Weighted Least Squares**: For each iteration, $\min \sum w_i (y_i - b - m t_i)^2$ is solved analytically.
3. **Robust Scale**: The median absolute deviation of residuals provides a robust scale estimate: $\hat{\sigma} = 1.4826 \cdot \text{median}(|r_i|)$.
4. **Huber Weights**: Weights are assigned as:

$$w_i = \begin{cases} 1 & \text{if } |r_i| \leq 1.345\,\hat{\sigma} \\ 1.345\,\hat{\sigma}/|r_i| & \text{otherwise} \end{cases} \tag{28}$$

Five informativeness gates then validate the result: (i) P90–P10 HFR variation must exceed $\max(0.15 \cdot \text{median}(y), 0.05)$; (ii) c must be bracketed within 10% margins of the scan range; (iii) the observed minimum must not be at scan endpoints; (iv) $m > 0$ (positive slope); (v) ≥ 5 samples retain weight ≥ 0.5 after outlier rejection.

The confidence score is:

$$C = \min\left(\frac{n_{\text{used}}}{n_{\min}}, 1\right) \times \max\left(1 - \frac{\text{RMSE}_{\text{rel}}}{0.35}, 0\right) \times \beta_{\text{bracket}} \tag{29}$$

where $\beta_{\text{bracket}} = 1.0$ if data exists on both sides of c, else 0.6.

8.4 ROI-Based Focus Metrics

Rather than computing a single HFR across the entire sensor, the `focus` module evaluates multiple Regions of Interest (ROIs) to detect field curvature and tilt:

The default configuration places 5 ROIs (center plus four corners), each sized at $\text{clamp}(\min(w, h)/4,\ 128,\ 512)$ pixels. Stars with peak values ≥ 0.98 of the saturation level are rejected. The aggregate HFR is the median across all valid ROIs (requiring ≥ 5 stars per ROI, ≥ 25 total).

8.5 Focus Monitor

The `astro-node` daemon includes a `focus_monitor` that continuously tracks optical focus quality via filesystem-watching:

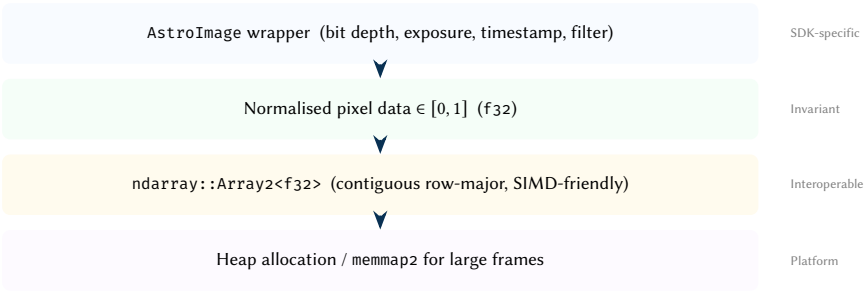


Figure 61. Image data representation layers. The SDK maintains the $[0, 1]$ normalisation invariant at the `AstroImage` boundary while allowing direct `ndarray` access for zero-copy interoperability with external processing libraries.

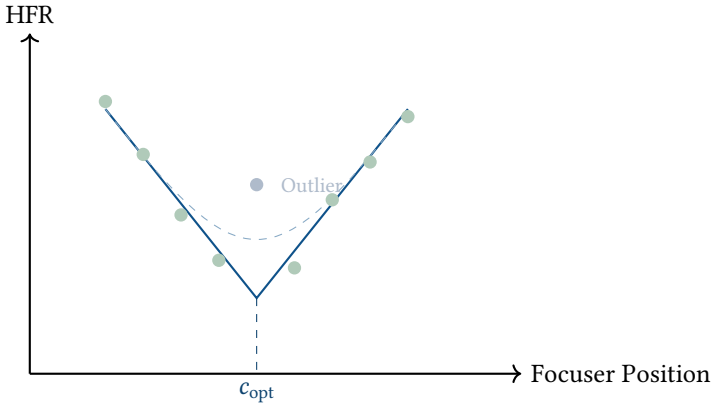


Figure 62. Autofocus V-Curve plotting HFR against focuser position. The robust IRLS fit suppresses the outlier data point.

The baseline HFR is computed as the trailing mean of the 30th percentile of the last N frames (configurable window). The degradation metric:

$$\delta = \left(\frac{\text{HFR}_{\text{current}}}{\text{HFR}_{\text{baseline}}} - 1 \right) \times 100\% \tag{30}$$

triggers `Drifting` at a configurable warn threshold and `Bad` (autofocus required) at the refocus threshold. A linear regression on the recent HFR window provides the trend slope for predictive refocus scheduling.

8.6 SCT Collimation

The `collimation` module analyses defocused star donuts from Schmidt-Cassegrain telescopes to guide mirror alignment [0]:

- Eccentricity vector computation from the donut shape.
- Azimuthal asymmetry mapping to detect coma direction.
- Translation of coma direction to adjustment screw instructions.

9 Live Stacking (EAA)

Electronically Assisted Astronomy (EAA) requires maintaining a continuously updated, high-precision image stack in real-time. The `live_stack` module utilizes Welford's online algorithm [0] to compute the running mean and variance without accumulating all frames in memory.

As each new frame x_n arrives, it is registered to the master and integrated into a 32-bit floating-point accumulator. Welford's algorithm prevents catastrophic cancellation errors that occur in naive sum-of-squares variance calculations, ensuring robust signal-to-noise ratio tracking over infinite sequence lengths.

10 Guiding & Periodic Error

The `guiding` module in `astro-core` provides tracking analysis and closed-loop offset calculations.

10.1 Periodic Error Analysis

Equatorial mounts driven by worm gears exhibit periodic tracking errors due to machining imperfections. The SDK implements a frequency-domain analysis using the Fast Fourier Transform (FFT) [0] to isolate the fundamental worm period and its harmonics.

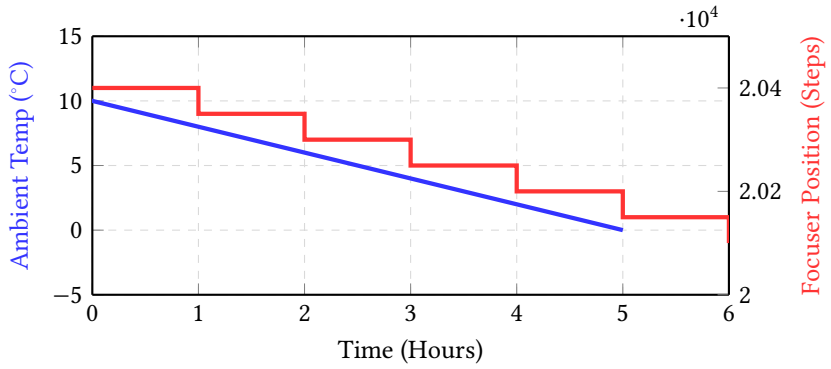


Figure 63. Temperature compensation loop. As the ambient temperature drops steadily (blue), the `astro-node` daemon autonomously injects computed inward micro-steps to the focuser (red) to maintain critical focus without requiring a sequence-interrupting V-Curve scan.

Table 2. Thermal expansion coefficients for common telescope materials.

Material	α (μm m^{-1} °C^{-1})
Aluminum	23.1
Carbon Fiber	0.5
Custom	User-defined

10.2 6-Term Mechanical Pointing Models

While polar alignment corrects the physical tracking axis, modern observatory mounts still suffer from mechanical flexure and non-orthogonal machining. The SDK implements a 6-term equatorial pointing model [0] (inspired by TPoint) to map raw encoder coordinates to true celestial coordinates.

By plate-solving a grid of stars across the sky, the system computes a least-squares fit for the 6 coefficients, allowing the `astro-node` orchestrator to perform closed-loop precision slews that place targets perfectly on the sensor without iterative centering.

10.3 The Meridian Flip Maneuver

A German Equatorial Mount (GEM) tracking a target from East to West will eventually run the camera and filter wheel into the physical pier. The `sequencer` must execute a Meridian Flip to continue tracking.

The robotic control logic computes the Hour Angle H continuously. When the target crosses the meridian ($H = 0$) plus an allowed safety margin, the orchestrator suspends the autoguider, flips the mount axes, plate-solves to confirm the new center, re-calibrates the guider vectors, and resumes the exposure sequence.

10.4 Alt-Az Field Rotation & Derotation

Unlike equatorial mounts, Alt-Az systems suffer from field rotation: the celestial field rotates relative to the sensor as a function of the target's position and the observer's latitude.

The `derotation` module calculates the instantaneous field rotation rate (ω_{rot}) in degrees per second. For hardware with integrated field derotators (e.g., Benro Polaris), the orchestrator issues continuous angular corrections to maintain astrometric alignment.

10.5 Dithering Strategies

To eliminate fixed-pattern noise (FPN) and sensor artifacts during stacking, the `guiding` module implements several dithering walks.

Three pattern generators are implemented:

1. **Fibonacci Spiral**: The golden-angle spiral $\theta_i = i \cdot 2\pi(\sqrt{5}-1)/2$, $r_i = s\sqrt{i+1}$ distributes dither offsets with optimal areal coverage. The $\sqrt{i}$ radius growth ensures equal density per annular ring.
2. **Random Walk**: A deterministic xorshift64 PRNG seeded with (seed + i) generates offsets in $[-s, +s]$ per axis. Determinism via the seed enables session reproducibility.
3. **Grid**: An expanding square lattice $(\lfloor i/k \rfloor - c,\ i \bmod k - c) \cdot s$ where $k = \lceil \sqrt{i} \rceil$ and $c = (k-1)/2$.

All patterns enforce a *radial clamp*: if the cumulative offset exceeds $r_{\max}$ (default 25″), the offset vector is scaled to lie on the bounding circle. This prevents the guide star from drifting out of the guide camera's field.

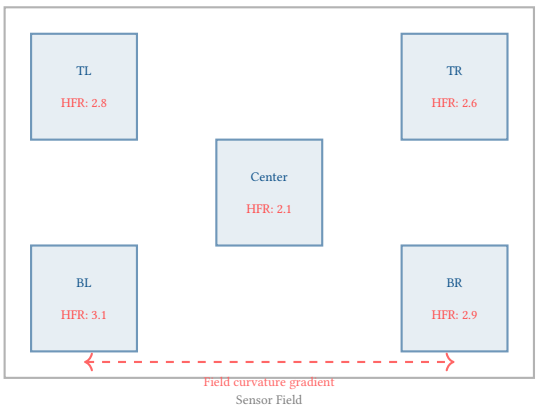


Figure 64. ROI-based focus evaluation. Five ROIs (center + four corners) independently measure HFR. A significant gradient between center and edge HFR values indicates field curvature or sensor tilt, triggering a tilt-correction advisory.

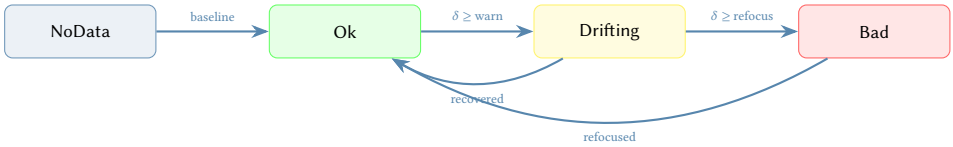


Figure 65. Focus monitor state machine. The degradation percentage $\delta = (\text{HFR}/\text{baseline} - 1) \times 100\%$ drives state transitions. An alert cooldown prevents repeated notifications.

10.6 PID Guide Controller

Autoguiding is fundamentally a closed-loop control problem: the guide camera measures the error between the guide star's current position and its reference position (the "lock" point), and the controller computes a corrective motor pulse to drive that error toward zero. The challenge is that the error signal is noisy (atmospheric seeing jitters the star at 1–3 Hz), the control actuator has latency (mount motors respond over 50–200 ms), and the update cadence is slow (typically one correction every 1–3 s, limited by exposure time). A well-tuned PID controller balances these constraints.

The `pid` submodule implements a discrete PID controller with independent RA and Dec axes. Each axis maintains its own integral accumulator and previous-error state. The continuous-domain control law is:

$$u(t) = k_p\, e(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de}{dt} \tag{31}$$

The three terms serve distinct physical roles in the guiding context:

- **Proportional** ($k_p = 0.5$): Corrects the instantaneous tracking error. A higher k_p gives faster response but risks oscillation as the mount overshoots and the next guide frame sees the opposite error.
- **Integral** ($k_i = 0.1$): Eliminates steady-state drift caused by polar misalignment or flexure. Without the I term, a constant drift produces a permanent offset that the P term alone cannot cancel. The integral accumulator is clamped (anti-windup) to prevent runaway during prolonged cloud gaps when the guide star is lost.
- **Derivative** ($k_d = 0.05$): Damps high-frequency oscillations by opposing rapid error changes. At the 1–3 s guide cadence, the D term primarily suppresses overcorrection ringing rather than tracking atmospheric turbulence (which is too fast to correct).

The default gains ($k_p = 0.5$, $k_i = 0.1$, $k_d = 0.05$) are conservative, prioritizing stability over aggressive convergence. Key safety features include:

- **Anti-windup**: The integral accumulator is clamped to $\pm 5''$, preventing unbounded growth during sustained errors (e.g., cloud passage).
- **Output clamp**: Total correction clamped to $\pm 10''$ to prevent dangerous large slews.
- **Deadband suppression**: If both RA and Dec corrections fall below $0.1''$, the entire pulse is suppressed, avoiding unnecessary motor activation that would introduce vibration.
- **Monotonic timestamps**: Non-monotonic timestamps are rejected with an error, preventing wraparound artifacts.

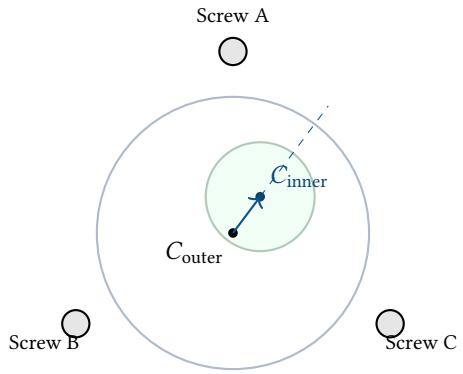


Figure 66. SCT collimation analysis. By computing the eccentricity vector between the outer bounds of the defocused star and the inner central obstruction, the system determines the axis of coma and computes the required secondary mirror screw adjustments.

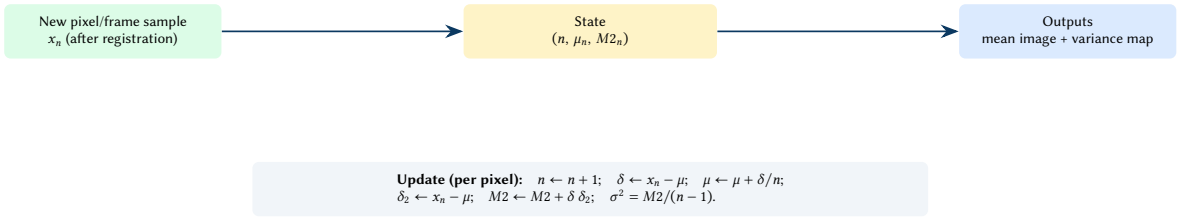


Figure 67. Welford live stacking as a streaming estimator. The system updates mean and variance online per pixel without retaining the full history.

10.7 Guide Calibration

Before autoguiding can commence, the system must determine the relationship between guide camera pixels and sky coordinates. The `calibration` submodule accumulates pulse–response samples and fits a linear model via ordinary least squares:

$$\alpha_{\text{axis}} = \frac{n\sum p_i d_i - \sum p_i \sum d_i}{n\sum p_i^2 - (\sum p_i)^2} \times s_{\text{plate}} \quad [''\,\text{ms}^{-1}] \tag{32}$$

where p_i is the pulse duration (ms), d_i is the observed pixel displacement, and s_{plate} is the plate scale. The calibration session proceeds in two phases: RA pulses first (minimum samples), then Dec pulses, producing per-axis sensitivity ($''\,\text{ms}^{-1}$), direction angle, R^2 confidence, and residual RMS.

10.8 Guide Quality Analysis

The `quality` submodule provides continuous monitoring of autoguiding performance through a multi-dimensional analysis pipeline:

1. **RMS Computation**: Separate east/north and combined total RMS in arcseconds: $\text{RMS}_{\text{total}} = \sqrt{\text{RMS}_E^2 + \text{RMS}_N^2}$.
2. **Settling Detection**: The system is declared settled when K consecutive samples (default 5) fall below the settling threshold (default $1.0''$). The time-to-settle metric is critical for optimizing post-dither wait times.
3. **FFT Periodic Error Spectrum**: The east-axis error signal is detrended (linear best-fit removed), windowed with a Hann function $w_i = 0.5(1-\cos(2\pi i/(n-1)))$, and transformed to produce a one-sided amplitude spectrum in arcseconds. A coherent gain correction compensates for the windowing energy loss.
4. **Polar Drift Fit**: Linear regression on the north-axis error yields the drift rate in $''\,\text{min}^{-1}$, directly diagnosing polar alignment error magnitude.

The composite quality score fuses all metrics into a single value $Q \in [0,1]$ that the sequencer uses to decide whether guiding is adequate for the current exposure. The design uses *multiplicative gating* rather than additive weighting: any single severe problem (unsettled mount, excessive dropouts) drives the score toward zero regardless of how good the other metrics are. This reflects the physical reality that a single catastrophic guiding failure ruins an entire sub-exposure.

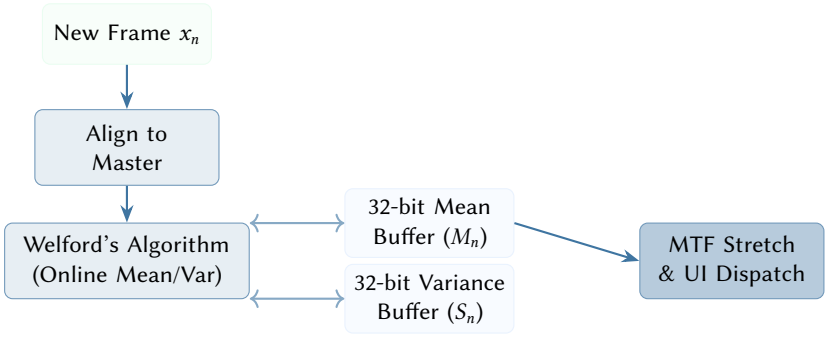


Figure 68. Live Stacking memory architecture. Welford's single-pass algorithm maintains numerical stability for the running 32-bit mean and variance buffers while preventing Out-of-Memory (OOM) errors during continuous hours-long EAA sessions.

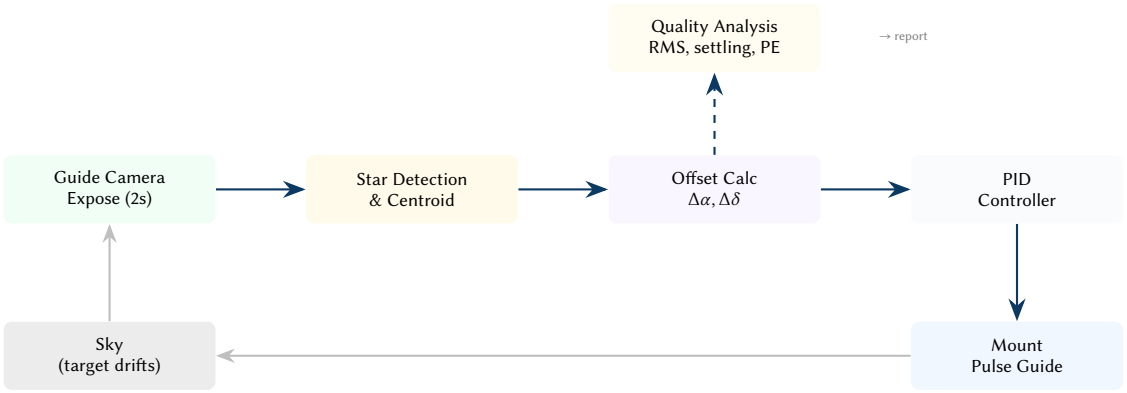


Figure 69. Closed-loop autoguiding architecture. The guide camera acquires frames at ~2 s cadence, star detection computes the centroid offset, and the PID controller issues pulse-guide corrections to the mount. The quality analysis module continuously monitors RMS, settling time, and periodic error via FFT.

$$Q = \text{clamp}\left(\frac{1}{1 + (\text{RMS}/1.0)^2} \times \gamma_{\text{settle}} \times \gamma_{\text{drops}} \times \gamma_{\text{spectral}},\ 0, 1 \right) \tag{33}$$

The base term is a sigmoid function of the RMS tracking error: $1/(1 + r^2)$ maps $0''$ RMS to $Q = 1.0$, $1''$ RMS to $Q = 0.5$, and $2''$ RMS to $Q = 0.2$, providing a smooth, continuous decay without hard thresholds. The three γ penalty factors then gate the score:

- $\gamma_{\text{settle}} = 0.7$ if the guider has not yet settled after a dither or slew (reduced from 1.0), reflecting the elevated risk of elongated stars during transient oscillations;
- $\gamma_{\text{drops}} = 1 - 0.3 \min(d/d_{\max}, 1)$ penalizes dropped guide frames proportionally—if 100% of frames are dropped, the score is reduced by 30%;
- $\gamma_{\text{spectral}} = 0.9$ if the FFT periodic error analysis or linear drift fit produces non-diagnostic results (insufficient data or degenerate spectrum).

As a worked example: at $0.8''$ RMS with a settled guider, no drops, and clean diagnostics, $Q = 1/(1 + 0.64) \times 1.0 \times 1.0 \times 1.0 = 0.61$—adequate for deep-sky imaging at moderate focal lengths. The same RMS with an unsettled guider drops to $Q = 0.61 \times 0.7 = 0.43$, which would trigger the sequencer to delay the next exposure until settling completes.

10.9 Guided Search Patterns

When starting a new session or recovering from a lost guide star, the `search` submodule generates systematic search patterns to locate the target field:

The search system features auto-expansion: after a configurable number of failures, the step size scales by 1.5× and the radius by 1.75×. Lock gating requires a plate-solve with ≥ 15 matched stars, RMS $\leq 2.5''$, and maximum residual $\leq 6.0''$ before declaring the search complete.

11 Lucky Imaging & Planetary Capture Optimization

For high-resolution planetary and lunar capture, the `astro-vision` crate implements Lucky Imaging [0]. Atmospheric turbulence, characterized by the Fried parameter r_0 [0], continuously distorts the incident

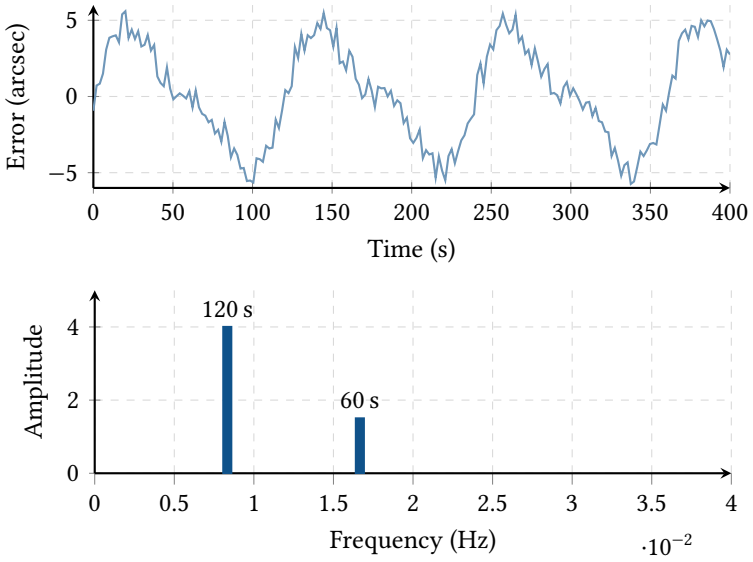


Figure 70. Top: Time-domain simulated tracking error exhibiting a fundamental 120-second worm period and noise. Bottom: FFT frequency-domain spectrum isolating the fundamental frequency (0.0083 Hz) and its first harmonic (0.0166 Hz).

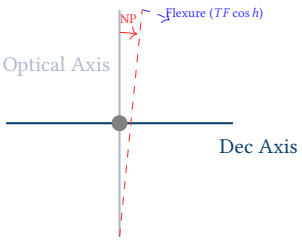


Figure 71. Mount Kinematics. The pointing model corrects for six fundamental mechanical errors: Index errors (IH, ID), Polar Elevation/Azimuth (ME, MA), Non-perpendicularity (NP), and Tube Flexure (TF).

wavefront. By capturing high-speed video and selecting only the sharpest frames, the pipeline effectively "freezes" the seeing.

11.1 Multi-Metric Frame Scoring

Rather than relying on a single sharpness indicator, the SDK computes three complementary metrics for each captured frame and fuses them into a weighted composite score:

$$S_{\text{frame}} = 0.3\,\hat{L} + 0.3\,\hat{T} + 0.4\,\hat{P} \tag{34}$$

where the individual metrics are:

1. **Laplacian Variance** ($\hat{L}$): The discrete Laplacian $L(x,y) = I(x{-}1,y)+I(x{+}1,y)+I(x,y{-}1)+I(x,y{+}1)-4\,I(x,y)$ is evaluated across all interior pixels and the variance of the resulting kernel response serves as the sharpness indicator. High-frequency detail (planetary belts, crater rims) produces large Laplacian variance.
2. **Tenengrad Gradient** ($\hat{T}$): Horizontal and vertical Sobel-like gradient magnitudes (g_x, g_y) are summed in quadrature across the frame:
$$T = \frac{1}{N_{\text{px}}} \sum_{x,y} \left(g_x^2 + g_y^2\right) \tag{35}$$
Values below 10^{-4} are clamped to suppress numerical noise on blank sky regions.
3. **Strehl Proxy** ($\hat{P}$): The ratio of the peak pixel to the local mean within a 5×5 neighborhood serves as a proxy for the Strehl ratio. Frames captured during isoplanatic "frozen seeing" moments produce pronounced peaks in this metric.

11.2 Planetary Disc Detection

Before derotation or stacking, the pipeline must locate the planetary disc within the sensor frame. The SDK uses a two-stage approach:

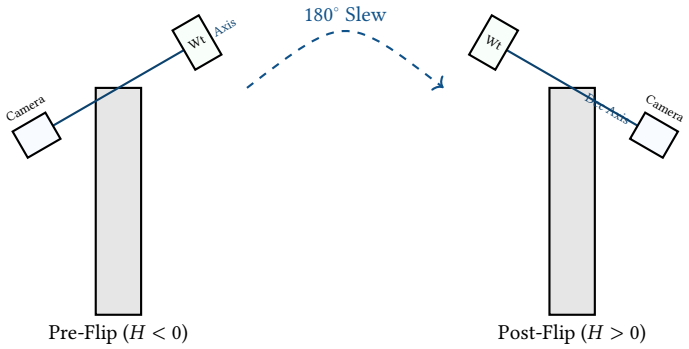


Figure 72. The Meridian Flip. To prevent collision with the pier, the orchestrator halts tracking, rotates the Right Ascension axis by 12 hours (180°), and reverses the Declination axis, effectively placing the camera "above" the mount.

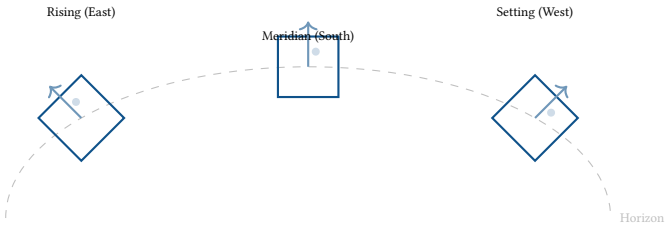


Figure 73. The field rotation effect on an Alt-Az mount. The target (represented by the sensor frame) rotates relative to the celestial field as it traverses the sky, requiring active mechanical derotation for long exposures.

1. **Threshold Centroiding**: Pixels above $I_{\min} + 0.1\,(I_{\max} - I_{\min})$ are accumulated via weighted first moments (m_{10}, m_{01}, m_{00}) to produce the initial centroid $(\bar{x}, \bar{y})$. The disc radius is estimated from the above-threshold area: $r = \sqrt{A/\pi}$.
2. **Spoke-Based Edge Refinement**: 32 radial spokes are cast from the centroid. Along each spoke, the maximum gradient magnitude identifies the true limb position. The median of the 32 limb distances yields a robust refined radius.

11.3 Planetary Derotation

Planets rotate on their axes, causing surface features to drift during a capture session. For Jupiter at opposition, the Great Red Spot moves at approximately $10''$ per minute. Two geometric models are provided:

11.3.1 Gnomonic (Spherical) Projection

For targets subtending more than a few arcminutes, the curvature of the planetary surface demands a proper spherical treatment. The SDK implements the gnomonic (central) projection:

$$\cos c = \sin\varphi_0 \sin\varphi + \cos\varphi_0 \cos\varphi \cos(\lambda - \lambda_0) \tag{36}$$

$$x' = \frac{\cos\varphi \sin(\lambda - \lambda_0)}{\cos c} \tag{37}$$

$$y' = \frac{\cos\varphi_0 \sin\varphi - \sin\varphi_0 \cos\varphi \cos(\lambda - \lambda_0)}{\cos c} \tag{38}$$

where (φ_0, λ_0) is the sub-observer point and (φ, λ) are surface coordinates. The inverse mapping reconstructs the spherical coordinates from projected pixel positions via $\rho = \sqrt{x'^2 + y'^2}$, $c = \arctan(\rho)$.

11.3.2 Orthographic Projection

For smaller angular extents or quick-look processing, the orthographic model provides a computationally cheaper alternative:

$$x' = \cos\varphi \sin\lambda, \qquad y' = \sin\varphi \tag{39}$$

with a visibility gate $x'^2 + y'^2 < 1$ to reject features on the far hemisphere.

11.3.3 Planetary Rotation Parameters

The derotator requires accurate rotation periods and pole orientations for each planet:

Insight:
The choice between gnomonic and orthographic projections is driven by the planet's angular diameter. For Jupiter (~ 45″ at opposition), the gnomonic model is essential; for Mars (~ 15″), orthographic suffices within sub-pixel registration error.

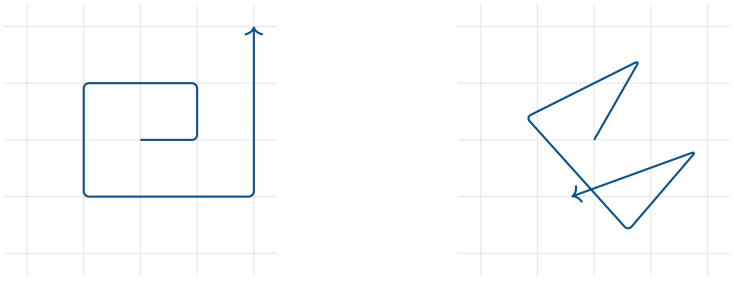


Figure 74. Dithering patterns supported by the `DitherGenerator`. The Spiral walk ensures uniform coverage of the neighborhood, while the Random walk minimizes periodic correlation with sensor-level defects.

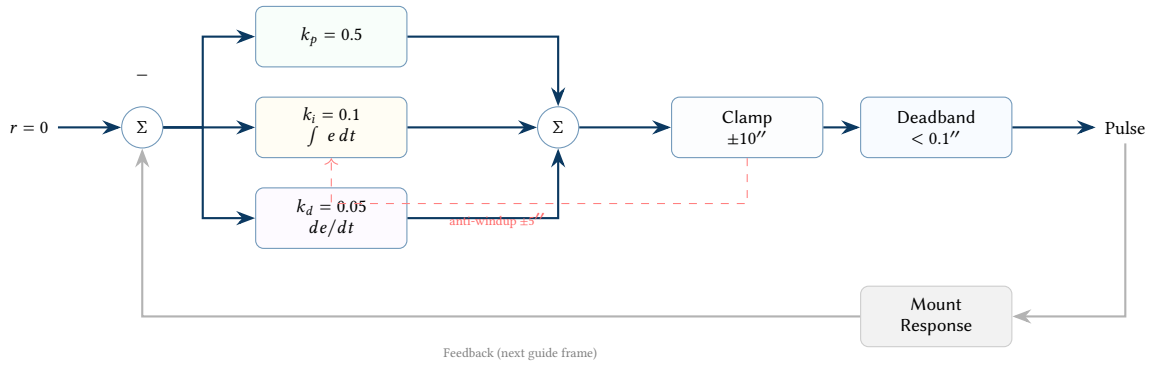


Figure 75. PID guide controller block diagram. The integral branch is clamped to ±5″ for anti-windup protection. Both RA and Dec corrections must exceed the 0.1″ deadband before a pulse is issued, suppressing sub-seeing jitter corrections.

The sub-observer longitude at any Julian Date is computed from the rotation rate:

$$\lambda_{\text{sub}}(t) = \frac{(t - t_{\text{J2000}}) \times 24}{P_{\text{rot}}} \times 360^{\circ} \pmod{360} \tag{40}$$

11.4 Maximum Untrailed Exposure

The derotation module also computes the maximum single-frame exposure time for alt-az mounts before field rotation trails a point source beyond a configurable pixel threshold:

$$t_{\max} = \frac{\Delta_{\max} \cdot s}{|\omega_{\text{rot}}|} \tag{41}$$

where $\Delta_{\max}$ is the acceptable trail length in pixels, s is the plate scale in arcsec px^{-1}, and the instantaneous field rotation rate for an alt-az mount is:

$$\omega_{\text{rot}} = \frac{15 \cos\phi \cos A}{\cos h} \quad [\text{″ s}^{-1}] \tag{42}$$

with observer latitude ϕ, target azimuth A, and altitude h.

11.5 Lucky Stacking Pipeline

The complete planetary capture pipeline proceeds:

> **Design Decision 10**
>
> Design Rationale: 64-bit Accumulation The final stacking pass uses `f64` accumulation rather than the source bit depth. When stacking 2000+ selected frames, 32-bit floating point would lose the least-significant bits of faint planetary detail (polar hood structure on Mars, Cassini division on Saturn). The extra 8 bytes per pixel is a negligible cost relative to the improved dynamic range.

12 Lunar & Solar Capture

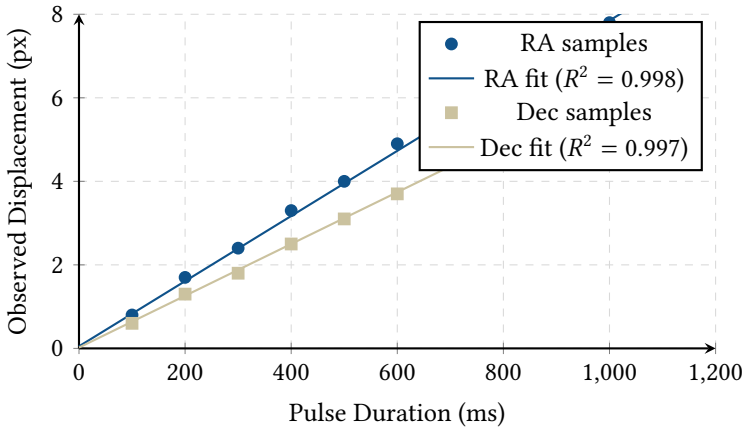


Figure 76. Guide calibration linear regression. The RA axis (blue) shows higher sensitivity than Dec (green) due to the cosine declination factor. The high R^2 values confirm linear mount response.

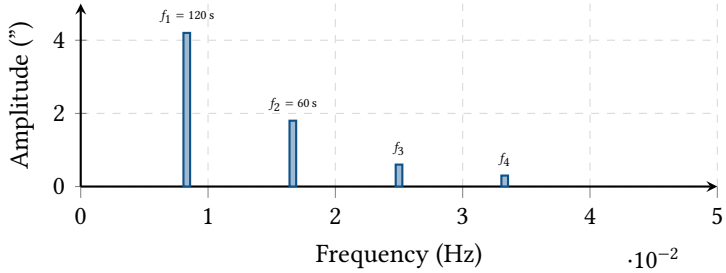


Figure 77. FFT periodic error spectrum from guide quality analysis. The fundamental worm period ($f_1 = 120$ s) and its harmonics reveal mechanical imperfections in the mount's drive train. This diagnostic enables targeted PEC correction.

The Moon and Sun present distinct imaging challenges compared to deep-sky targets: extreme dynamic range (the full Moon at $m \approx -12.6$ versus the surrounding sky), rapid angular motion ($\sim 0.5'' \, \mathrm{s}^{-1}$), and for the Sun, mandatory optical filtration.

12.1 Lunar Ephemeris & Phase Model

The `lunar` module implements the Meeus perturbation series (Chapter 47 [0]) with ten principal longitude terms and five latitude terms. The five fundamental arguments—mean longitude L', mean elongation D, solar mean anomaly M, lunar mean anomaly M', and argument of latitude F—are evaluated as fourth-order polynomials in Julian centuries T from J2000:

$$L' = 218.^\circ 316 + 481267.^\circ 881 \; T \tag{43}$$

$$D = 297.^\circ 850 + 445267.^\circ 111 \; T \tag{44}$$

$$M = 357.^\circ 529 + 35999.^\circ 050 \; T \tag{45}$$

$$M' = 134.^\circ 963 + 477198.^\circ 868 \; T \tag{46}$$

$$F = 93.^\circ 272 + 483202.^\circ 018 \; T \tag{47}$$

The illumination fraction k quantifies how much of the lunar disc is sunlit, ranging from $k = 0$ (new Moon) to $k = 1$ (full Moon). This is the primary driver for deep-sky planning: a Moon with $k > 0.5$ significantly elevates the sky background in broadband filters, making narrowband filters essential. The illumination is derived from the phase angle i, which is the angular distance between the Moon and the anti-solar point as seen from Earth:

$$k = \frac{1 + \cos i}{2}, \qquad i = 180^\circ - D - 6.^\circ 289 \sin M' + 2.^\circ 1 \sin M - 1.^\circ 274 \sin(2D - M') \tag{48}$$

The correction terms account for the Moon's orbital eccentricity ($6.289^\circ \sin M'$), the Earth's orbital eccentricity ($2.1^\circ \sin M$), and the evection perturbation ($1.274^\circ \sin(2D - M')$). Without these corrections, the predicted illumination can err by up to ±8%—enough to misjudge whether a session should use broadband or narrowband filters.

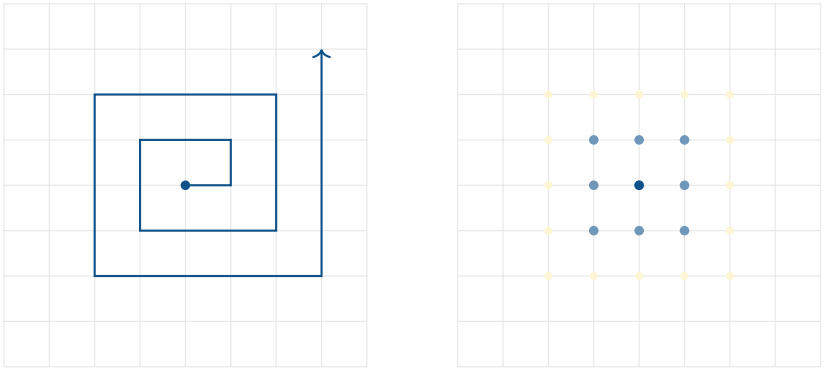


Figure 78. Guided search patterns for initial target acquisition. The spiral pattern (left) covers the field in a single continuous path; the concentric grid (right) searches in expanding rings, optimizing for targets near the expected position.

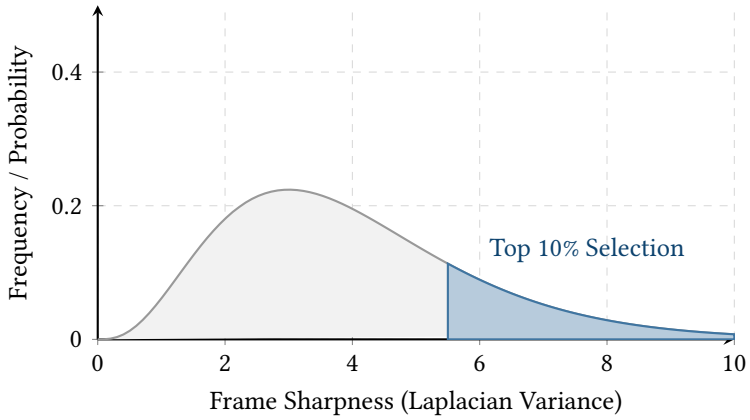


Figure 79. Frame sharpness distribution in a Lucky Imaging sequence. The highly skewed right tail represents the rare frames captured during moments of negligible atmospheric turbulence, which are retained for stacking.

12.2 Eclipse Detection & Timeline Prediction

For lunar eclipses, the SDK computes the Earth's shadow geometry by placing the umbral and penumbral cones opposite the Sun. The shadow radii at the Moon's distance are modeled as fixed angular values derived from the mean Earth–Moon–Sun geometry:

$$r_{\text{umbra}} = 0.^{\circ}74, \qquad r_{\text{penumbra}} = 1.^{\circ}28, \qquad r_{\text{moon}} = 0.^{\circ}26 \tag{49}$$

The eclipse magnitude measures the fraction of the Moon's diameter immersed in the umbral shadow:

$$\text{mag}_{\text{umbral}} = \frac{r_{\text{umbra}} + r_{\text{moon}} - \sigma}{2\, r_{\text{moon}}} \tag{50}$$

where σ is the angular separation between the Moon's center and the shadow center. The magnitude has a direct physical interpretation: mag < 0 means no umbral contact (penumbral eclipse only), $0 <$ mag < 1 indicates a partial eclipse, and mag ≥ 1 indicates totality—the entire disc is within the umbra. The sequencer uses these thresholds to automatically adjust exposure bracketing and filter selection during each eclipse phase.

Eclipse type classification follows:

The timeline prediction module performs a grid search at 1-minute resolution across a ±720 minute window centered on the approximate eclipse time, computing the angular separation at each step and identifying the seven standard contact times: P1 (penumbral first contact), U1–U4 (umbral contacts), and P4 (penumbral last contact).

Insight:
Eclipse timeline prediction enables the sequencer to pre-program filter and exposure changes: broadband RGB during partial phases, narrowband Hα during totality (to capture the "blood moon" chromatic signature), and high-cadence lucky imaging for Baily's Beads at second and third contacts.

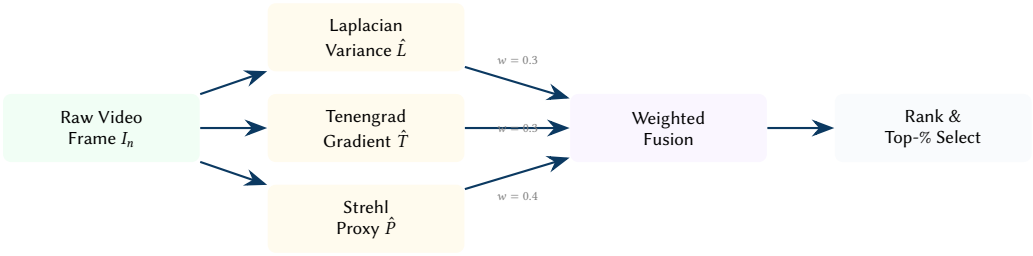


Figure 80. Lucky imaging multi-metric scoring pipeline. Three independent sharpness indicators are fused with configurable weights; the Strehl proxy receives the highest weight due to its direct correlation with atmospheric coherence.

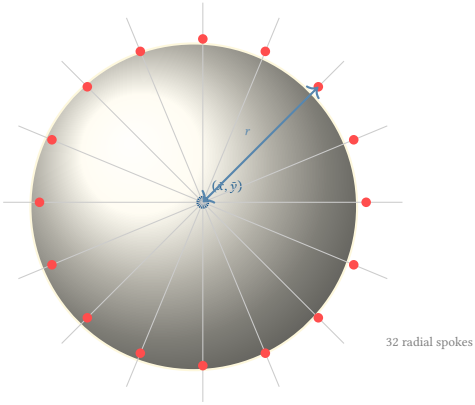


Figure 81. Spoke-based planetary disc detection. After threshold centroiding, 32 radial spokes locate limb-edge gradient maxima (red dots). The median spoke distance yields a robust disc radius estimate.

12.3 Planetary Ephemeris

The `planetary` module implements a truncated VSOP87D heliocentric ecliptic series for the five naked-eye planets. Kepler's equation is solved via Newton–Raphson iteration (10 iterations max):

$$E_{n+1} = E_n - \frac{E_n - e \sin E_n - M}{1 - e \cos E_n} \tag{51}$$

The geocentric equatorial position is obtained by subtracting the Earth's heliocentric vector and rotating by the J2000 obliquity $\varepsilon = 23°439$:

$$x_{\text{eq}} = g_x \tag{52}$$
$$y_{\text{eq}} = g_y \cos \varepsilon - g_z \sin \varepsilon \tag{53}$$
$$z_{\text{eq}} = g_y \sin \varepsilon + g_z \cos \varepsilon \tag{54}$$

Angular diameters are computed from the inverse-distance law using reference diameters at 1 AU (Table 3).

13 Sequencer State Machine

The `sequencer` module implements a deterministic, event-driven state machine that orchestrates multi-target imaging sessions. All state transitions are pure functions of the current state and incoming events, with no internal timers or I/O—making the sequencer fully testable and reproducible.

13.1 Event-Driven Transitions

The sequencer processes five event categories with strict priority ordering:

1. **SafetyChanged**: Highest priority. `Unsafe` transitions any state to `Aborted`; `Marginal` pauses imaging; `Safe` resumes from pause.
2. **FrameCaptured**: Updates filter progress, checks dither interval, evaluates meridian flip trigger ($H > 6$ h).
3. **FrameRejected**: Increments rejection counter; if transparency drops below the per-target threshold, transitions to `CloudInterference`.
4. **SlewComplete / GuiderSettled / FlipDone**: Hardware acknowledgments that advance the main acquisition flow.

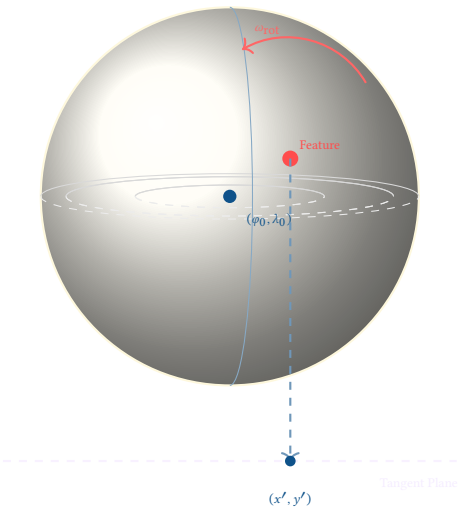


Figure 82. Gnomonic derotation geometry. A surface feature on the rotating planet is projected onto the tangent plane at the sub-observer point. As the planet rotates (angular rate ω_{rot}), the projected position changes non-linearly due to limb foreshortening.

Table 3. Planetary rotation parameters used for derotation.

Planet	P_{rot} (h)	**Pole RA** (°)	**Pole Dec** (°)	$d_{1\,\text{AU}}$ (")
Jupiter	9.925	268.057	64.495	196.74
Saturn	10.656	40.589	83.537	165.60
Mars	24.623	317.681	52.886	9.36
Venus	5832.6	272.76	67.16	16.69
Mercury	1407.6	281.01	61.42	6.73

5. **InitiateShutdown**: Graceful shutdown sequence (collect flats → park mount → close shutter).

13.2 Target Selection Algorithm

When the sequencer returns to `Idle` (target complete, cloud skip, or session start), it executes a multi-stage filtering and ranking pipeline to select the next observation target. The algorithm is designed to maximize scientific yield while respecting physical and environmental constraints.

The pipeline proceeds through six stages:

1. **Status filter**: Only targets with status `Pending` are considered. Targets marked `InProgress`, `Complete`, `Skipped`, or `Failed` are excluded.
2. **Time window**: Targets with a `not_before_jd` or `not_after_jd` constraint are rejected if the current Julian Date falls outside the allowed window. This supports scheduling of time-critical events (e.g., a comet at perihelion, or a galaxy only visible in the first half of the night).
3. **Coordinate transform**: Each surviving target's equatorial coordinates (α, δ) are converted to horizontal (A, a) using the observer's current local sidereal time and latitude.
4. **Horizon mask gate**: The target's altitude must exceed $\max(a_{\min}, \texttt{mask}(A))$ where $a_{\min}$ is the per-target minimum altitude (typically 30°) and $\texttt{mask}(A)$ is the azimuth-dependent horizon profile from trees, buildings, or mountains.
5. **Lunar separation**: Targets whose angular distance from the Moon is less than `min_moon_sep_deg` (default 45°) are rejected to avoid scattered moonlight contamination, particularly important for broadband imaging.
6. **Ranking**: Surviving candidates are sorted by `TargetPriority` (High > Normal > Low), with ties broken by current altitude—preferring targets closer to the meridian where atmospheric extinction is minimized and tracking error is lowest.

If no target survives all filters, the sequencer remains in `Idle` and re-evaluates on the next `TimeTick` event, as targets near the eastern horizon may rise above the altitude gate within minutes.

Figure 83. End-to-end Lucky Imaging pipeline for planetary capture. Frames are scored, ranked, derotated for planetary rotation, registered to sub-pixel accuracy via bilinear interpolation, and averaged with 64-bit accumulation.

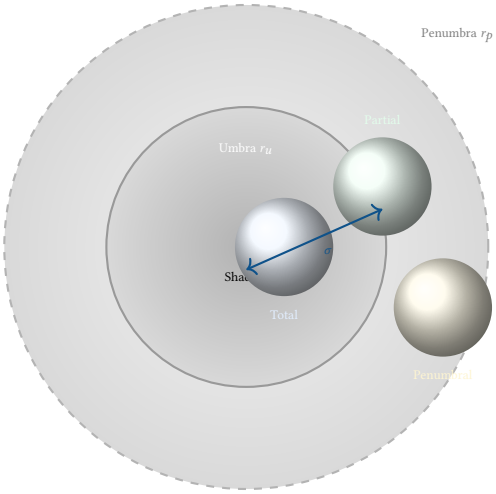


Figure 84. Lunar eclipse geometry. The Moon's position relative to the umbral and penumbral shadow cones determines the eclipse type and magnitude. The SDK predicts all seven contact times (P1, U1, U2, mid, U3, U4, P4) by grid-searching ±720 minutes around the approximate opposition time.

13.3 Holy Grail Timelapse Integration

The sequencer optionally integrates the `exposure_solver` for Holy Grail timelapse sequences. After each frame capture, the solver recomputes the optimal camera settings:

$$\Delta\text{EV} = \log_2\left(\frac{\text{ADU}_{\text{target}}}{\text{ADU}_{\text{current}}}\right) \tag{55}$$

During twilight ($-18^\circ < h_\odot < 5^\circ$), the maximum per-frame adjustment is clamped to ±1.5 stops to prevent visible flicker; during full darkness, the clamp tightens to ±0.5 stops. The solver preferentially adjusts shutter speed before touching ISO, minimizing read-noise penalties.

13.4 Timeline Planning

The `timeline` submodule projects an entire session across a Julian Date window, producing a sequence of `TimelineSlot` entries (Gap, Slew, Settle, Imaging, FocusCheck, MeridianFlip):

13.5 Terminal User Interface

The `astro-node` TUI provides a real-time operational view of the sequencer, guider, and safety systems in a terminal-based layout designed for headless observatory operation over SSH.

14 Mosaic Planning

The `mosaic` module in `astro-core` generates multi-panel imaging grids for targets larger than a single sensor field of view.

14.1 Grid Generation

Panel centres are computed in the tangent plane with configurable overlap fraction (default 15%). The generator handles two numerical subtleties:

1. **RA wrapping.** Right ascension values near 0h/24h boundary are kept in unwrapped form relative to the grid centre during computation, then normalised to $[0^\circ, 360^\circ)$ on output.
2. **Pole proximity.** At $|\delta| \geq 85^\circ$, RA convergence makes the flat-sky approximation unreliable. The module flags affected grids with a reduced confidence score.

14.2 Path Ordering

Panels are ordered in a serpentine (boustrophedon) pattern to minimise total slew distance [0]: odd rows are traversed left-to-right, even rows right-to-left.

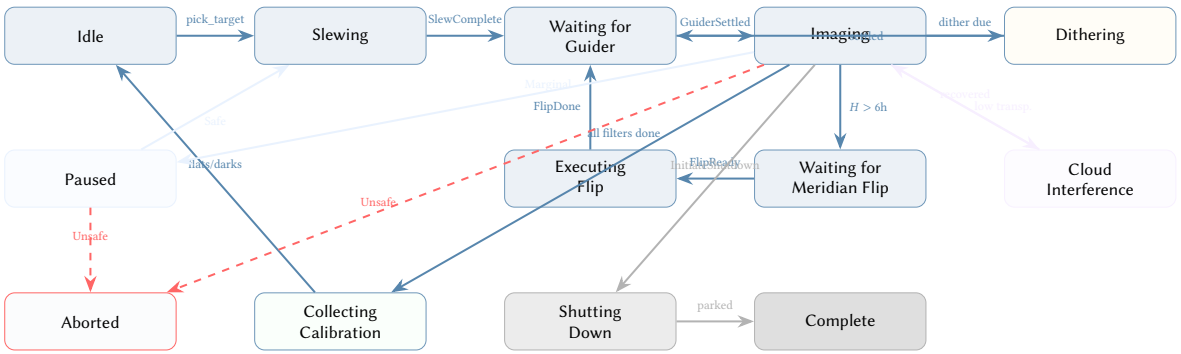


Figure 85. Sequencer state machine. All transitions are event-driven; safety events (Unsafe, Marginal) pre-empt all other processing. The Cloud Interference hold state preserves target progress while waiting for conditions to improve.

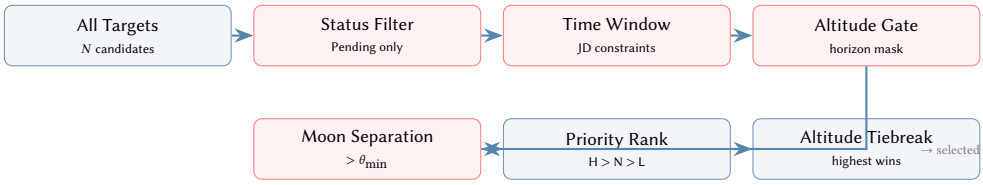


Figure 86. Target selection pipeline. Red stages are eliminative filters that reduce the candidate set; blue stages rank the survivors.

14.3 Coverage Tracking

Each panel progresses through a lifecycle: `Planned` → `Captured` → `Stacked`. The coverage module rasterises panel footprints in the tangent plane using WCS transforms to compute overlap regions and overall completeness.

15 Testing & Validation

The SDK contains over 2,600 unit tests embedded in the source files via Rust's `#[cfg(test)]` convention. Tests fall into three categories:

1. **Reference-value tests.** Coordinate transforms, sidereal time, and airmass are validated against published tables (Meeus worked examples, USNO data). Tolerances are set at the level of the algorithm's intrinsic accuracy (e.g. 1 arcsec for GMST, 0.01 for airmass).
2. **Round-trip tests.** Equatorial→horizontal→equatorial and `pixel_to_sky`→`sky_to_pixel` round-trips are verified to sub-arcsecond precision.
3. **Property tests.** Stacking and registration use synthetic frames with known transforms and injected outliers to verify that sigma-clipping rejects the outliers and the recovered transform matches the ground truth.

Verifying the Lunar Ephemeris

The lunar position module is validated against the Meeus worked example for 1992 April 12 at 0h TDT: the expected geocentric ecliptic longitude is 133°10′, latitude −3°13′, and distance 368,410 km. The test asserts agreement within 0.5° for position and 500 km for distance—within the Meeus series truncation error.

16 Hardware-in-the-Loop Simulation

The `astro-sim` crate provides a complete digital twin of the observatory hardware. Rather than relying on physical clear skies for integration testing, the framework employs Hardware-in-the-Loop (HITL) simulation [0] to validate the orchestration logic under adverse conditions.

The simulator applies physical constraints including maximum slew rates, meridian flip geometry, and tracking errors. It utilizes a pseudo-random star catalog to render synthetic FITS frames on-the-fly, integrating Gaussian PSFs with Poisson-distributed photon noise and Gaussian read noise, allowing the vision pipeline to run end-to-end against realistic data.

16.1 Virtual Rig: Component Hierarchy

To accurately model complex multi-camera setups (e.g., dual-Sony co-axial imaging), the `astro-sim` digital twin utilizes a hierarchical component tree.

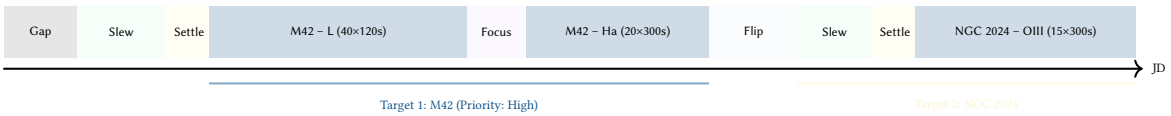


Figure 87. Session timeline projection showing the sequencer's planned slot allocation across a night. Slew, settle, and meridian flip overheads are explicitly modeled to produce accurate integration-time estimates.

Figure 88. TUI mockup of the `astro-node` terminal interface. The four-panel layout shows sequencer progress (top-left), guider tracking with real-time RA/Dec error plot (top-right), safety and fleet status (bottom-left), and an event log (bottom-right). The interface is designed for headless operation over SSH.

17 Edge AI Safety Engine

Automated observatory operations require robust environmental safety monitoring. The `astro-sentinel` crate evaluates telemetry and imaging feeds through a boolean rules engine, optionally backed by ONNX machine learning models [0] for computer vision tasks.

By executing lightweight convolutional neural networks directly on the edge (e.g., Raspberry Pi), the system detects encroaching cloud cover or wildlife near the mount without requiring an external internet connection or cloud-based API calls.

17.1 Observatory Fault-Tolerance Statechart

Managing distributed hardware requires formally verifying the failure states. The orchestrator is modeled as a hierarchical state machine (Statechart) [0] to guarantee that no sequence of asynchronous hardware errors can leave the physical equipment in a vulnerable state.

Rust's asynchronous ecosystem (`tokio`) natively supports cancellation. By wrapping the active imaging loop in a `tokio::select!` block alongside the Sentinel event channel, a weather interrupt instantly drops the imaging future, aborts the camera exposure, and initiates the deterministic Park & Close procedure.

17.2 Damage Detection & Frame Triage

The `damage` submodule within `astro-sentinel` evaluates each captured frame for transient contamination and quantifies the impact on the final stack's signal-to-noise ratio. The growing density of satellite constellations (Starlink, OneWeb) means that a typical 300-second deep-sky exposure at mid-latitude has a 10–30% probability of containing at least one satellite trail, making automated detection and triage essential for unattended operation.

The damage model computes a weighted affected-area fraction:

$$A_{\text{affected}} = \sum_{\text{class } c} w_c \cdot p_c \tag{56}$$

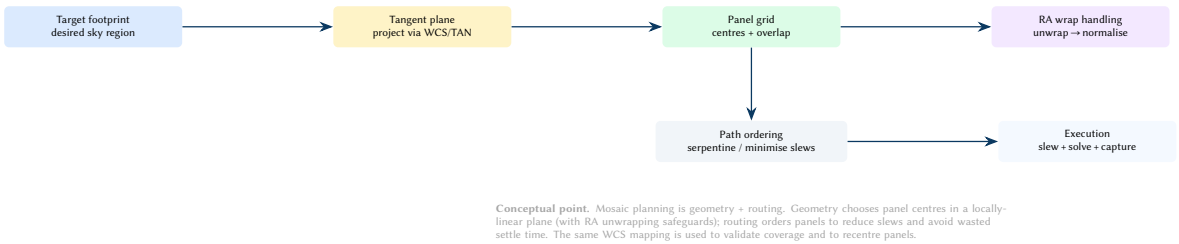


Figure 89. Mosaic planning as coverage geometry plus path planning. Panel centres are computed in the tangent plane and then ordered into an efficient traversal.

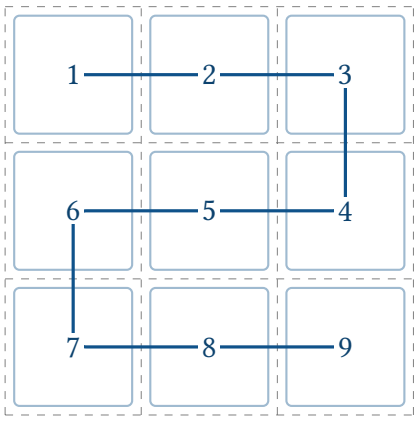


Figure 90. A 3 × 3 mosaic grid demonstrating the boustrophedon (serpentine) path ordering to minimize mount slew distance.

where w_c is the fractional area weight per classification type and p_c is the detection confidence. The weights reflect the typical spatial footprint of each contaminant:

- **Satellites/aircraft** ($w = 0.05$): Linear trails typically affect 3–8% of pixels in a frame, and are difficult to remove via sigma-clipping if they recur in similar positions across multiple subs.
- **Transients** ($w = 0.02$): Cosmic ray hits and meteor flashes affect isolated pixel clusters ($< 1\%$ of frame area) and are well-handled by the stacking outlier rejector.

The estimated SNR loss converts the fractional area into a decibel penalty:

$$\Delta\text{SNR} = -10 \log_{10}(1 - A_{\text{affected}}) \quad [\text{dB}] \tag{57}$$

This formula captures the non-linear relationship between contaminated area and quality: a 5% affected area costs only 0.22 dB (negligible), but 25% costs 1.25 dB (a meaningful loss equivalent to discarding one quarter of the integration time), and 50% costs 3 dB (half the signal lost). The configurable thresholds translate these losses into automated decisions:

17.3 IR Cloud Detection

The `ir_cloud` classifier operates on thermal infrared (8–14 μm) all-sky camera imagery, exploiting the thermal contrast between warm cloud bases and the cold clear sky. Unlike optical cloud detection (which fails in moonless conditions), thermal IR provides reliable detection regardless of ambient illumination.

The detection proceeds in four stages:

1. **Binary Thresholding**: Each pixel with normalized intensity > 0.35 is flagged as a potential cloud pixel. This threshold corresponds approximately to the thermal signature of low-altitude cumulus or stratus at typical mid-latitude observatory sites. Clear sky at zenith typically registers below 0.15 in normalized thermal intensity.
2. **Connected Component Analysis**: A stack-based (non-recursive) flood fill with 4-connectivity (north, south, east, west neighbors) labels contiguous regions of flagged pixels. The stack-based implementation avoids recursion depth limits on large all-sky images (commonly 640 × 480 or 1024 × 768). Boundary checking uses wrapping subtraction to safely handle edge pixels without underflow.
3. **Blob Size Filtering**: Connected components smaller than 400 pixels are rejected as sensor noise, hot pixels, or isolated warm objects (aircraft, birds). This threshold is calibrated to the typical angular

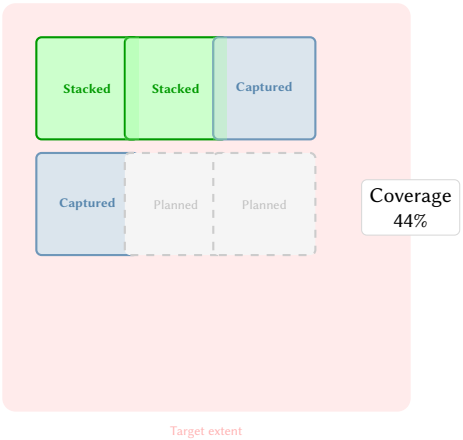


Figure 91. Mosaic coverage tracking with panel lifecycle states. Green panels are fully stacked, blue panels are captured but not yet stacked, and grey dashed panels are planned. Overlap regions (darker green) provide registration anchors between adjacent panels.

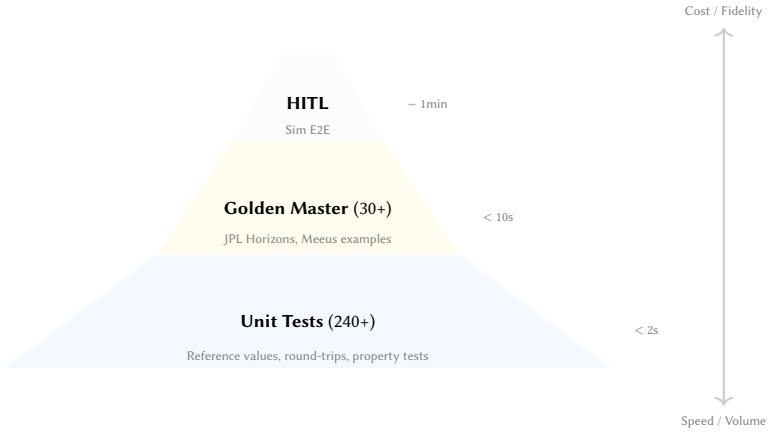


Figure 92. Test pyramid for `astro-core`. The base consists of 2,600+ fast unit tests (reference-value, round-trip, and property-based). The middle layer validates against astronomical gold standards. The apex uses HITL simulation for full end-to-end orchestration testing.

resolution of consumer all-sky IR cameras, where 400 pixels subtends approximately 5° of sky—below which a "cloud" is unlikely to affect imaging.

4. **Confidence Scoring**: The final confidence combines two complementary indicators:

$$c = \text{clamp}\left(2\, f_{\text{cloud}} + \frac{b_{\max}}{2000},\ 0.1,\ 0.99\right) \tag{58}$$

where f_{cloud} is the fraction of the sky covered by cloud pixels and $b_{\max}$ is the pixel count of the largest single blob. The cloud fraction term dominates for widespread overcast conditions; the blob size term elevates confidence when a single large cloud mass threatens the imaging target even if total coverage is low.

The confidence floor of 0.1 ensures that no frame is ever classified as "perfectly clear" (accounting for sensor noise and calibration uncertainty), while the ceiling of 0.99 prevents the system from ever being fully certain of total overcast (a small hole may exist outside the camera's field of view).

17.4 Rules Engine

The `rules` submodule provides a declarative, TOML-serializable trigger system that decouples detection logic from response policy. Each rule specifies a classification label (or transient event type), a minimum confidence threshold, and a trigger action.

Two action types are supported:

- **CaptureBurst**$\{n\}$: Immediately captures n frames at the current exposure settings, used for transient events (meteors, satellite flares) that benefit from rapid multi-frame documentation.

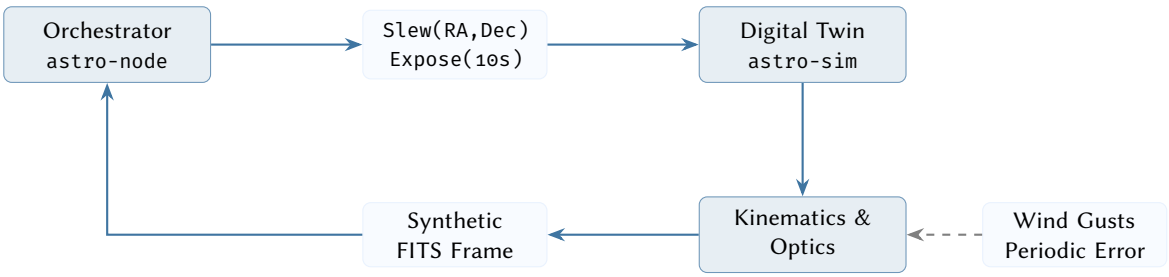


Figure 93. Hardware-in-the-Loop (HITL) closed-loop control simulation. The orchestrator interacts with a digital twin that models telescope kinematics and synthetically renders star fields with injected mechanical tracking errors.

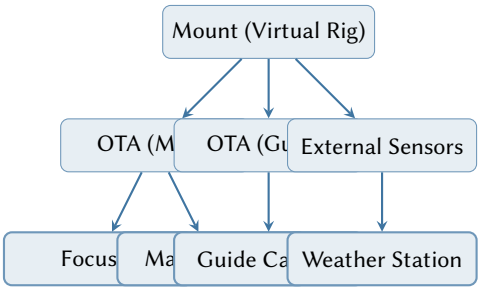


Figure 94. The hierarchical digital twin structure in `astro-sim`. Transformations (e.g., slewing the mount) propagate through the child nodes, ensuring that all optical paths maintain physical consistency relative to the mounting axis.

- **Notify**{message}: Emits a structured notification to the observer (via push notification, dashboard alert, or webhook), used for conditions requiring human judgment (e.g., "Wildlife detected near telescope—verify before resuming").

Rules are evaluated sequentially against all classifications from the current frame; every rule whose label matches a detection and whose confidence threshold is met contributes its action to the output. Actions are accumulated in rule-definition order and dispatched as a batch, allowing a single frame to trigger both a burst capture and a notification simultaneously.

The rule set is validated at load time: confidence thresholds must lie in $[0, 1]$, rule labels must be non-empty, and duplicate targets (same label and event type) are rejected to prevent accidental double-triggering.

18 Panorama Planning

The `panorama` module generates serpentine waypoint grids for wide-field panoramic capture, accounting for spherical geometry and mount limitations.

18.1 Waypoint Generation

Given the equipment parameters (focal length, sensor dimensions) and desired field coverage, the module computes the single-frame FOV:

$$\theta_x = 2\arctan\left(\frac{w_{\text{sensor}}}{2\,f_L}\right), \qquad \theta_y = 2\arctan\left(\frac{h_{\text{sensor}}}{2\,f_L}\right) \tag{59}$$

The step size between adjacent panels accounts for the user-specified overlap fraction η (typically 0.2–0.3):

$$\Delta_x^{\text{base}} = \theta_x(1-\eta), \qquad \Delta_y = \theta_y(1-\eta) \tag{60}$$

18.2 Spherical Cosine Correction

At higher altitudes, azimuthal convergence compresses the angular spacing. The module applies a cosine correction per row:

$$\Delta_x(\text{alt}) = \frac{\Delta_x^{\text{base}}}{\max(\cos(\text{alt}),\ 0.1)} \tag{61}$$

The 0.1 floor prevents division-by-zero near the zenith and ensures panels always overlap even at extreme altitudes.

The serpentine ordering reverses the column traversal direction on odd-numbered rows, minimizing the total slew distance. The output `PanoramaPlan` contains an ordered sequence of `PanoramaWaypoint` entries, each specifying the azimuth, altitude, row index, and column index.

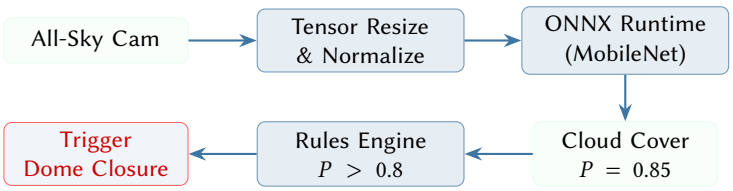


Figure 95. The Sentinel Safety Engine pipeline. Real-time imagery is processed via a cross-platform ONNX inference runtime. The resulting classification probabilities are evaluated against user-defined TOML rules to trigger emergency hardware shutdowns.

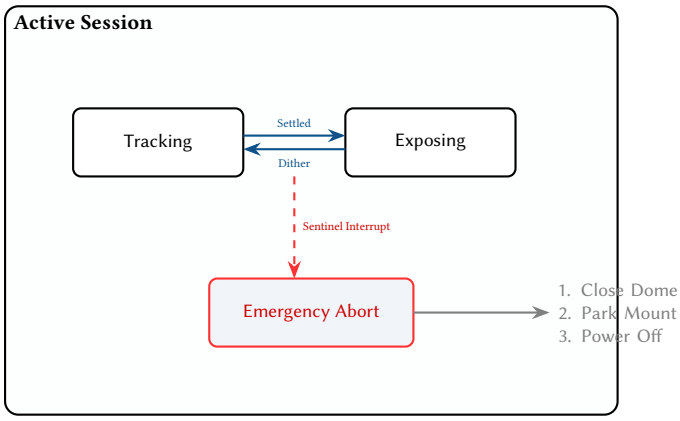


Figure 96. Simplified Harel Statechart of the observatory orchestrator. If the Edge AI Sentinel issues an interrupt (e.g., cloud detected) while the system is nested inside the Exposing or Tracking states, the hierarchy immediately preempts all active futures and transitions to the guaranteed Emergency Abort recovery sequence.

19 Fleet Coordination

The `astro-node` crate supports multi-rig observatory setups through a peer-to-peer fleet coordination protocol. Each node discovers its peers via mDNS and exchanges state via UDP broadcast.

19.1 Discovery & State Exchange

Each broadcast packet contains the node's hostname, current target, exposure status, battery level, safety state, and dither-readiness flag. The fleet manager maintains a peer hash map updated on each received packet.

19.2 Synchronized Dithering

When multiple rigs share a pier or are mechanically coupled (e.g., piggyback configurations), a dither move on one mount physically disturbs all co-mounted instruments. The fleet synchronization protocol ensures that dithering only occurs during inter-exposure gaps when no peer is actively integrating.

The critical coordination primitive is the `can_dither` predicate. A node may only issue a dither move when one of two conditions holds: (1) the fleet is empty (standalone operation), or (2) *all* peers report both `is_exposing = false` *and* `ready_to_dither = true`. This conservative AND-gate ensures that even if exposure durations differ across rigs, no dither occurs until the last peer finishes its current integration.

The fleet state is refreshed every 2 seconds via UDP broadcast, meaning the worst-case dither latency is one broadcast interval after the slowest peer completes its exposure. For typical deep-sky exposures of 60–300 s, this 2-second overhead is negligible ($< 3\%$ duty cycle loss).

The complementary `is_any_peer_unsafe` query implements a fleet-wide safety propagation: if *any* peer's sentinel detects an unsafe condition (cloud cover, wildlife, dew threshold), all peers receive the `is_unsafe = true` flag on the next broadcast cycle and can preemptively enter protective mode—even if their own local sensors have not yet detected the threat. This provides early warning for spatially correlated hazards (approaching weather fronts) that may reach one rig seconds before its neighbors.

20 Resiliency & Unified State (v0.11)

The v0.11 milestone addresses the architectural gap between high-level mission intent and low-level hardware safety. It introduces a unified mission schema and a centralized observatory state engine.

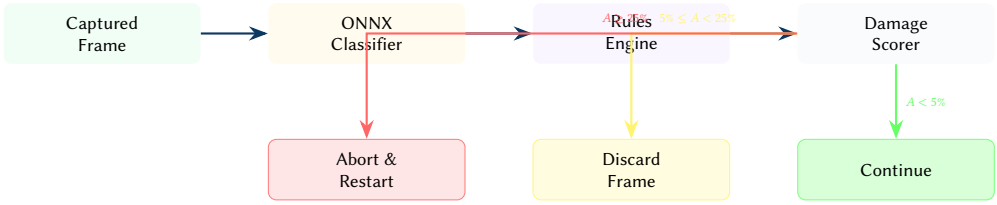


Figure 97. Damage detection pipeline. Each frame is classified, evaluated against the TOML rules engine, and scored. The mitigation decision (Continue, Discard, or Abort) is determined by configurable area thresholds.

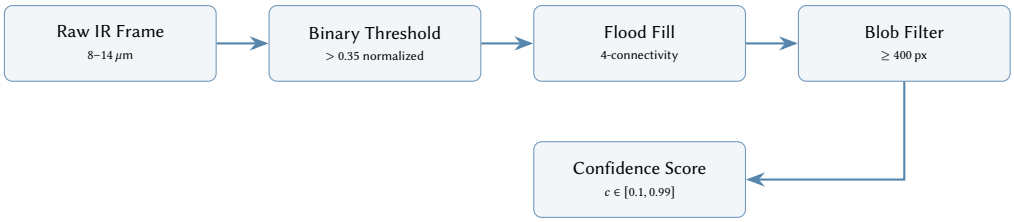


Figure 98. IR cloud detection pipeline. The four-stage algorithm converts a thermal all-sky image into a scalar confidence score suitable for safety decisions.

20.1 Unified Mission Schema

A critical requirement for professional-grade automation is consistency across planning and execution layers. The `mission` module in `astro-core` defines a versioned, serializable `Mission` struct that unifies the data model used by the TUI (`astro-node`), GUI (`astro-dashboard`), and the mobile FFI.

The `Mission` struct encapsulates the complete session specification: observer location, equipment profile, an ordered list of `SequenceTarget` entries (each with coordinates, capture plan, time constraints, and priority), global safety thresholds (minimum altitude, moon separation, transparency gate), and optional calibration directives (flat/dark frame collection at session end). By consolidating all session parameters into a single serializable document, the schema eliminates the class of bugs where the TUI and GUI produce subtly different sequencer inputs.

> **Design Decision 11**
>
> Schema Versioning & Migration The `Mission` schema includes a `version` field that enables forward-compatible evolution. When the sequencer loads a mission file, the `upgrade_json` method inspects the version tag and applies a chain of migration transforms—adding new fields with safe defaults, renaming deprecated keys, converting legacy coordinate formats—before deserializing into the current internal representation. This ensures that mission files saved by older SDK versions continue to execute correctly on newer nodes without manual editing, a critical requirement for field-deployed observatories that may not be updated simultaneously.

20.2 Autonomous Observatory State Engine

The `observatory` module implements a high-level state machine that wraps the imaging sequencer. It manages the transitions between functional states (`Startup`, `Ready`, `Imaging`, `Pausing`) and safety states (`Closing`, `Safe`).

20.3 WASM-Based Extensibility

To allow for dynamic extension without compromising core stability, the SDK utilizes a WASM-based plugin architecture. This enables third-party developers to contribute specialized `Sentinel` rules and hardware drivers that run in a high-performance, memory-safe sandbox.

21 Advanced Intelligence & Collaborative Science (v0.12)

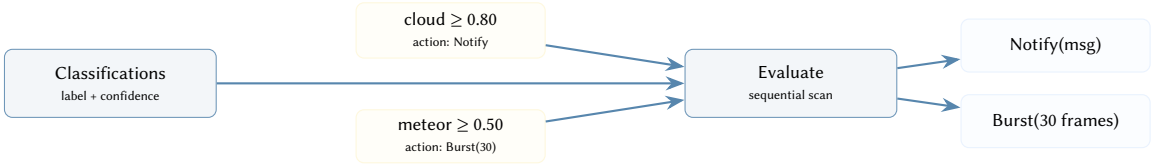


Figure 99. Rules engine evaluation. Classifications from the detection pipeline are matched against the rule set; all rules whose label and confidence criteria are satisfied produce actions that are accumulated and dispatched.

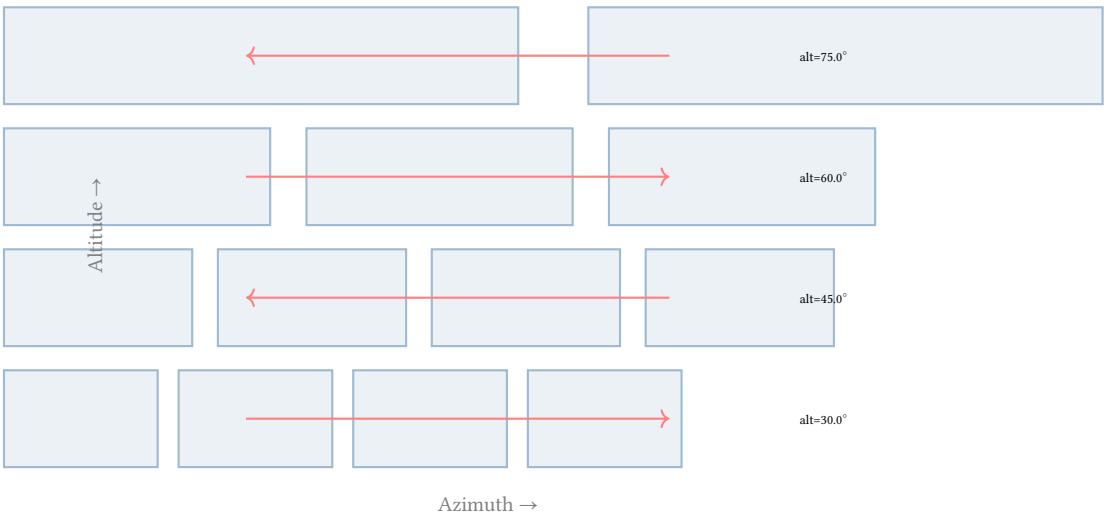


Figure 100. Panorama waypoint grid with cosine altitude correction. Higher-altitude rows require fewer panels as the azimuthal extent contracts. The serpentine path (red arrows) minimizes total slew distance by alternating direction on odd rows.

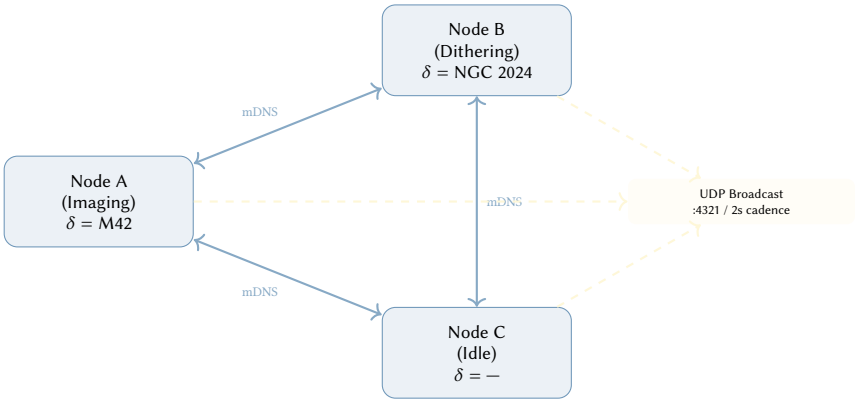


Figure 101. Fleet peer discovery. Each `astro-node` instance registers via mDNS (`_astro._tcp.local.`) and broadcasts its state (MessagePack-encoded) over UDP port 4321 every 2 seconds.

The v0.12 milestone elevates the SDK from a single-node controller to a collaborative intelligence engine, introducing global telemetry and autonomous optical health correction.

21.1 Collaborative Fleet Intelligence

Building on the fleet coordination logic (section 19), the v0.12 intelligence layer elevates the fleet from synchronized operation to collaborative science. Nodes exchange structured `FleetEvent` messages that carry semantic information about detected phenomena, enabling multi-site follow-up of transient events.

Two event types are supported:

- **Heartbeat**{uptime_secs}: Periodic liveness signal indicating that the node is operational. Peers use heartbeat absence (no update within 3 broadcast cycles = 6 s) to detect node failures.
- **TransientDetected**{label, coord, confidence}: A real-time alert carrying the classification label (e.g., “meteor”, “satellite flare”, “nova candidate”), the equatorial coordinates of the detection, and the classifier’s

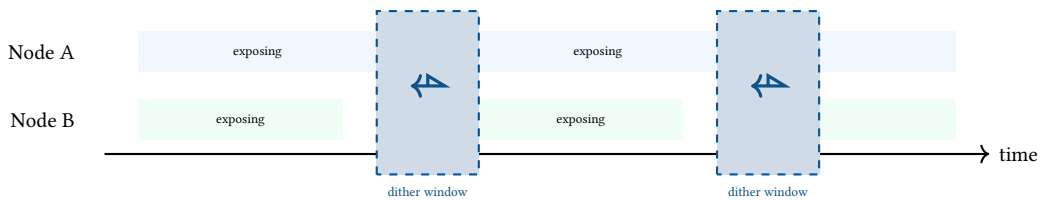


Figure 102. Synchronized dithering timeline for a two-node fleet. Dither moves (orange windows) only occur when *both* nodes report `is_exposing = false` and `ready_to_dither = true`.

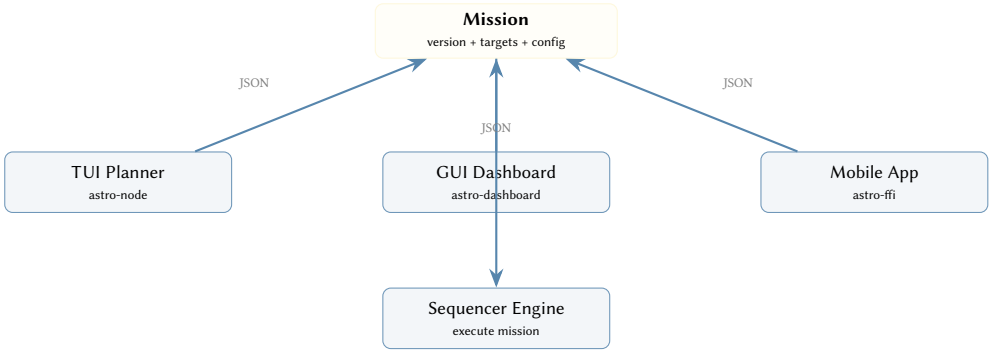


Figure 103. The Mission schema serves as the single source of truth across all planning interfaces. Each frontend serializes to the same JSON schema; the sequencer deserializes and executes identically regardless of the originating interface.

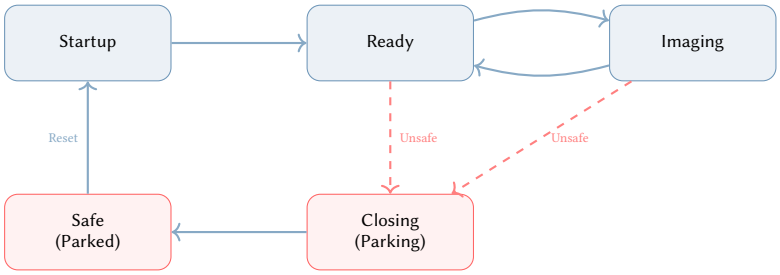


Figure 104. The Unified State Engine. The engine prioritizes safety constraints over mission execution. An `Unsafe` environment trigger (weather, power, or heartbeat loss) forces an immediate transition to `Closing` regardless of the current imaging state.

confidence score. Subscribing peers can automatically interrupt their current schedule to slew and capture the event from a different geographic vantage point.

The multi-vantage capture has scientific value beyond documentation: meteors observed from sites separated by 10–50 km provide parallax baselines sufficient to determine the meteor's altitude and atmospheric entry angle. The fleet event system distributes the coordinates so that each peer's sequencer can independently compute the local horizontal coordinates and verify that the target is above its own horizon mask before committing to the slew.

21.2 Distributed Sky Network (DSN)

The `DsnClient` enables a global, privacy-preserving atmospheric telemetry network. Local measurements of seeing (arcsec) and transparency are anonymized via coordinate quantization (eq. (62)) and submitted to a central registry.

$$(\lambda', \phi') = (\lfloor 10\lambda + 0.5 \rfloor / 10,\ \lfloor 10\phi + 0.5 \rfloor / 10) \tag{62}$$

This 0.1° rounding (approx. 11 km) protects user location privacy while providing sufficient resolution for a real-time global "Live Seeing Map" to aid in observational planning.

21.3 Closed-Loop Optical Correction

The `collimation` module in `astro-core` closes the loop between field aberration analysis and physical sensor adjustment. Given an error vector (dx, dy) in pixels representing the centroid offset between the actual and ideal optical axis, the module computes the optimal single-screw adjustment for a standard 3-screw collimator (screws spaced 120° apart).

21.3.1 Screw Projection Algorithm

For each of the three screws at angular positions $\theta_k = \theta_{\text{base}} + k \cdot 120°$ (for $k = 0, 1, 2$), the error vector is projected onto the screw's axis:

$$p_k = dx \cos\theta_k + dy \sin\theta_k \tag{63}$$

The screw with the largest absolute projection $|p_k|$ is selected as the one most aligned with the error direction. Its adjustment magnitude in fractional turns is:

$$f = \frac{p_{k^*}}{s} \tag{64}$$

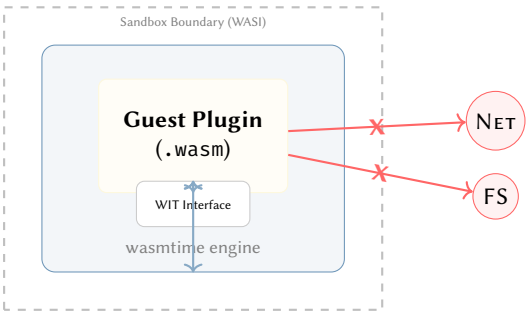


Figure 105. The WASM Plugin Sandbox. Guest plugins are strictly isolated from the host file system and network. All communication occurs over a strongly-typed WIT interface, ensuring that a faulty plugin cannot crash the primary observatory controller.

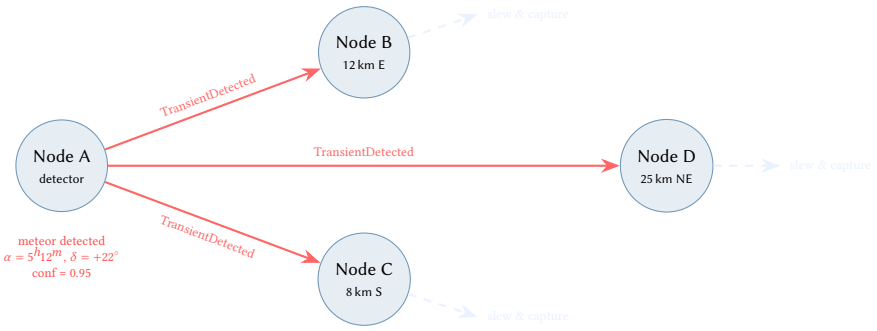


Figure 106. Transient event broadcast. Node A detects a meteor and broadcasts a `TransientDetected` event with equatorial coordinates. Peers B, C, and D autonomously slew to capture the event from different baselines, enabling triangulation and multi-perspective documentation.

where s is the calibrated pixels-per-turn constant (determined during initial setup, typically 30–50 pixels per full turn depending on the focuser and tilt plate mechanics).

21.3.2 Iterative Convergence with Damping

Mechanical backlash, flexure, and non-linear screw response make single-step correction unreliable. The system therefore operates in an iterative feedback loop (`SmartCollimation` state):

1. Capture a short test exposure and analyze field aberrations (HFR gradient across 5 ROIs).
2. Compute the error vector and optimal screw adjustment.
3. Apply only 70% of the calculated correction (damping factor $\gamma = 0.7$) to prevent overshoot.
4. Recapture and re-analyze. If the total error $\sqrt{dx^2 + dy^2}$ is below the target threshold, accept the correction. Otherwise, repeat from step 2.

The 70% damping factor was empirically determined to provide monotonic convergence within 3–4 iterations for typical amateur tilt plates, without the oscillatory behavior observed with $\gamma > 0.85$.

> **Design Decision 12**
>
> Design Rationale: UDP over TCP Fleet state exchange uses unreliable UDP broadcast rather than TCP connections. The rationale: (1) state packets are idempotent—a missed packet is superseded by the next broadcast 2 seconds later; (2) UDP broadcast requires zero connection management, simplifying dynamic fleet membership; (3) the 2-second cadence provides adequate temporal resolution for dither synchronization (typical exposures are 60–300 s). MessagePack serialization keeps packets compact (< 200 bytes per peer).

22 Projective Invariant Plate Solving

While sub-exposure registration relies on affine-invariant triangles, blind astrometric calibration (plate solving) must account for projective distortion across wide fields of view. The SDK utilizes the geometric hashing algorithm formalized by Astrometry.net [0].

The algorithm extracts four-star “quads” from the image. The two most distant stars define a normalized coordinate system, and the positions of the two internal stars yield a 4-dimensional hash code (c_x, c_y, d_x, d_y).

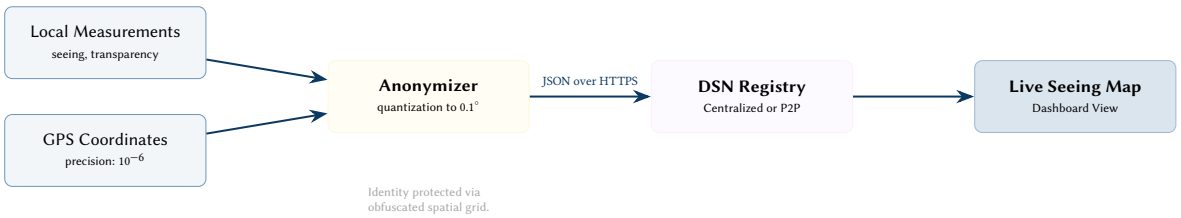


Figure 107. The Distributed Sky Network data flow. Precise site coordinates are never transmitted. The local node quantizes its location to an 11 km grid before submitting atmospheric telemetry, ensuring that the global network gains intelligence without compromising individual privacy.

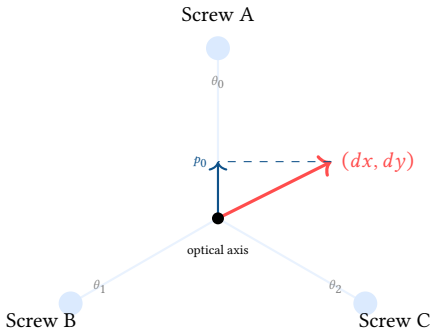


Figure 108. Three-screw collimator geometry. The error vector (dx, dy) is projected onto each screw axis; the screw with the largest projection magnitude receives the adjustment.

This geometric fingerprint is matched against a pre-compiled k-d tree index of the sky, yielding robust astrometric solutions even in the presence of optical distortion.

23 Binary Protocol Codecs

Interfacing with modern tracking hardware often requires reverse-engineering proprietary communication layers. The `polaris-proto` crate implements a robust, asynchronous binary codec for the Benro Polaris smart tracker.

The protocol parser guarantees memory safety and resilience against fragmented TCP packets by utilizing Tokio's asynchronous `Decoder` trait. It buffers incoming bytes in a state machine, verifying the frame header, length bounds, and computing a cyclic redundancy check (CRC-16) before dispatching the strongly-typed payload struct to the upper layers.

The actual wire format uses a 10-byte header:

Table 4. Polaris protocol header structure (10 bytes total).

Field	Size	Description
`payload_len`	4 bytes (u32)	Payload size (max 1 MiB)
`kind`	2 bytes (u16)	Frame type discriminant
`request_id`	4 bytes (u32)	Request–response correlation

The codec enforces a maximum payload size of 1 MiB (configurable) and returns typed errors (`FrameTooLarge`, `TruncatedPayload`) for malformed input. The `request_id` field enables multiplexing of concurrent commands over a single TCP connection.

23.1 Exposure Intelligence: Holy Grail Ramp

The `exposure_solver` implements flicker-free timelapse exposure ramping for sunset-to-stars sequences. The EV-shift model:

$$\Delta\text{EV} = \log_2\left(\frac{\text{ADU}_{\text{target}}}{\max(\text{ADU}_{\text{current}},\ 1)}\right) \tag{65}$$

The solver preferentially adjusts shutter speed before modifying ISO to minimize read-noise degradation. When increasing exposure: shutter is lengthened first (up to the configured maximum), then ISO is raised. When decreasing: ISO is lowered first (to the minimum), then shutter is shortened.

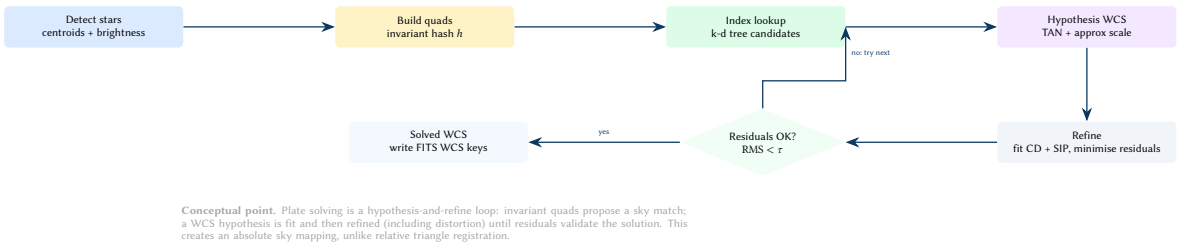


Figure 109. Plate solving as a decision DAG. Quads propose hypotheses via index lookup; refinement fits WCS parameters; residuals accept/reject candidates.

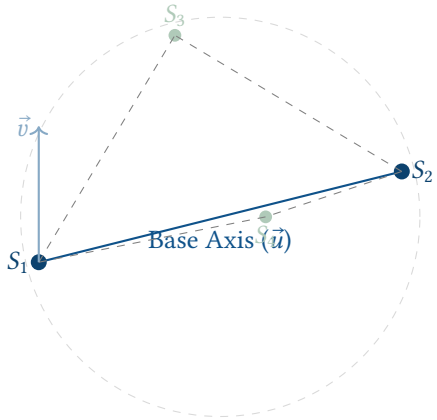


Figure 110. A 4-star geometric "quad" used for blind plate solving. The two most distant stars (S_1, S_2) define a normalized 2D coordinate system ($\vec{u}, \vec{v}$). The internal positions of S_3 and S_4 within this system create a translation-, rotation-, and scale-invariant 4D hash code used to rapidly search the celestial index.

24 Camera Sensor Noise Characterization

The `noise_model` module in `astro-vision` provides a first-principles signal-to-noise model encompassing all dominant noise sources in CMOS and CCD astronomical imaging.

24.1 Noise Budget Decomposition

For a single sub-exposure of duration t seconds, the total noise variance is the quadrature sum of four independent sources:

$$\sigma^2_{\text{total}} = S + n_{\text{sky}}\, t + n_{\text{dark}}\, t + \sigma^2_{\text{read}} \tag{66}$$

where S is the signal (Poisson shot noise), n_{sky} is the sky background flux ($\text{e}^-\ \text{s}^{-1}\ \text{px}^{-1}$), n_{dark} is the dark current ($\text{e}^-\ \text{s}^{-1}\ \text{px}^{-1}$), and σ_{read} is the read noise in electrons.

The SNR for n stacked sub-exposures is:

$$\text{SNR} = \frac{n \cdot S}{\sqrt{n \cdot (S + n_{\text{sky}}\, t + n_{\text{dark}}\, t + \sigma^2_{\text{read}})}} \tag{67}$$

24.2 Sky Background Estimation

The sky background is the dominant noise source for deep-sky imaging under anything other than pristine dark skies. Under Bortle 5 suburban conditions, the sky contributes more electrons per pixel per second than a 14th-magnitude galaxy—meaning the camera is mostly photographing light pollution, with the target signal buried underneath. Understanding the sky flux is therefore critical for determining optimal sub-exposure times: the exposure must be long enough for the sky signal to dominate the read noise ("sky-limited" regime) but short enough to avoid saturation.

The sky flux per pixel is modeled from the Bortle scale surface brightness (μ in mag arcsec^{-2}):

$$n_{\text{sky}} = 10^{(m_0 - \mu)/2.5} \cdot \pi(D/2)^2 \cdot s^2 \cdot \eta_{\text{QE}} \tag{68}$$

Each factor has a direct physical interpretation:

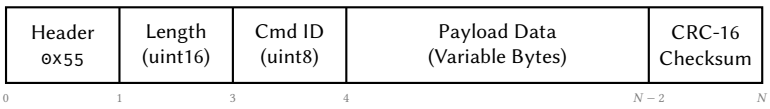


Figure 111. Binary packet frame layout for the Polaris protocol. The codec module implements a state machine using the `tokio-util` `Decoder` trait, buffering raw TCP stream bytes until a complete, CRC-validated frame is accumulated.

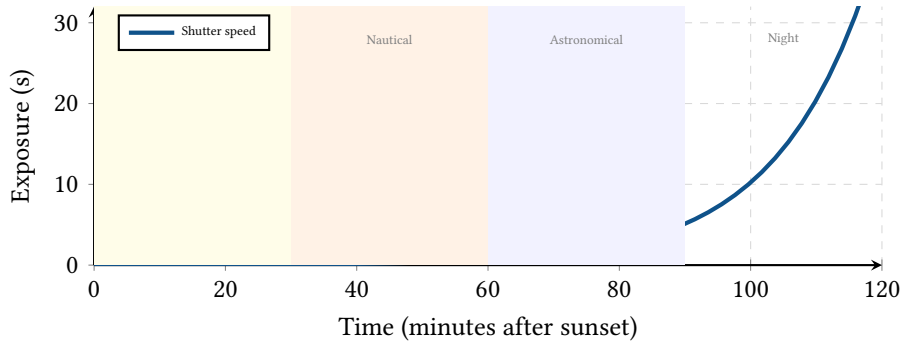


Figure 112. Holy Grail timelapse exposure ramp. During the rapid twilight transition, the solver permits up to ±1.5 stops per frame to track the falling sky brightness. After astronomical twilight, the rate limit tightens to ±0.5 stops to prevent visible flicker in the final video.

- $10^{(m_0-\mu)/2.5}$ converts the surface brightness from magnitudes to a linear flux density, referenced to the zero-point $m_0 = 26.5$ (the magnitude of a source producing 1 photon per second through a 1 m^2 aperture);
- $\pi(D/2)^2$ is the telescope aperture collecting area—faster optics gather more sky photons per pixel, making the sky *brighter* in each pixel even though the target is also brighter;
- s^2 is the solid angle subtended by one pixel in arcsec^2—larger pixels capture more sky background per pixel;
- η_{QE} is the sensor's quantum efficiency, converting photons to detected electrons.

As a concrete example: a Bortle 5 sky ($\mu = 20.0$) through an $f/5$ 200 mm aperture telescope with 1.0"/px plate scale and 80% QE produces ~ 4.2 electrons/pixel/second of sky background. The same setup under Bortle 2 ($\mu = 21.8$) drops to ~ 0.9 e$^-$/px/s—a 4.7× improvement that directly translates to deeper images for the same total integration time.

Table 5. Bortle scale surface brightness values.

Bortle Class	μ (mag arcsec^{-2})	Description
B1	22.0	Excellent dark site
B2	21.7	Typical dark site
B3	21.3	Rural sky
B4	20.8	Rural/suburban
B5	20.3	Suburban
B6	19.5	Bright suburban
B7	19.0	Suburban/urban
B8	18.5	City
B9	17.5	Inner city

24.3 Seeing-Limited SNR

For point sources under atmospheric seeing, the optimal photometric aperture has radius $r_{\text{ap}} = 1.67 \cdot \text{FWHM}/2$. The noise integral over this aperture includes contributions from $A = \pi r_{\text{ap}}^2$ pixels:

$$\text{SNR}_{\text{seeing}} = \frac{S \cdot n \cdot t}{\sqrt{S\, n\, t + A \cdot n \cdot (\sigma_{\text{read}}^2 + n_{\text{dark}}\, t + n_{\text{sky}}\, t)}} \tag{69}$$

24.4 Photon Transfer Curve

The `noise_model` includes a Photon Transfer Curve (PTC) fitting routine for empirical camera characterization. From flat-frame pairs at various exposure levels, the variance–mean relationship is:

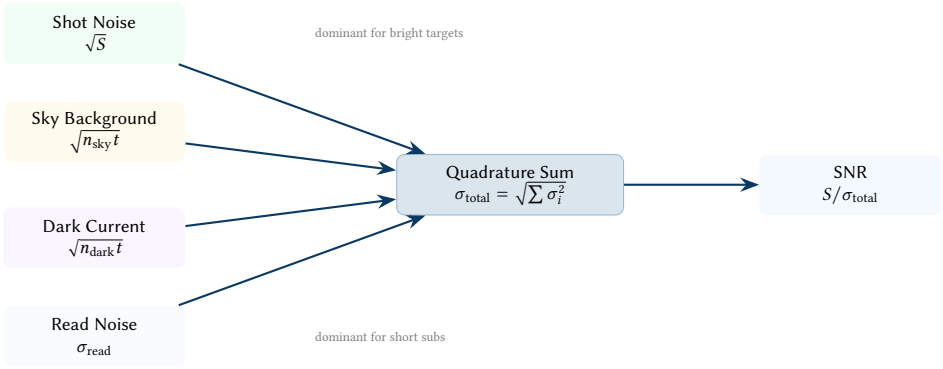


Figure 113. Noise budget decomposition. Each noise source contributes independently; the quadrature sum yields the total noise floor. For deep-sky imaging under light-polluted skies, the sky background typically dominates; for short planetary exposures, read noise dominates.

$$\sigma^2_{\text{ADU}} = \frac{1}{g} \cdot \bar{I}_{\text{ADU}} + \frac{\sigma^2_{\text{read}}}{g^2} \tag{70}$$

Linear regression yields the sensor gain $g = 1/\text{slope}$ ($\text{e}^-\ \text{ADU}^{-1}$) and read noise $\sigma_{\text{read}} = g \cdot \sqrt{\text{intercept}}$.

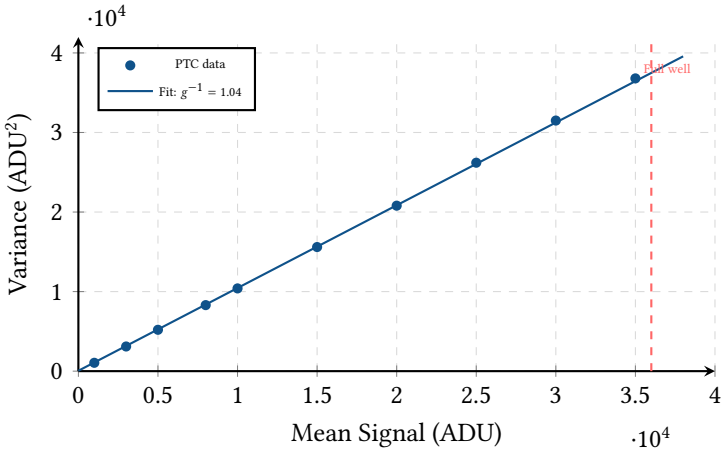


Figure 114. Photon Transfer Curve. The linear region yields the sensor gain ($g = 1/\text{slope}$) and read noise (from the y-intercept). Departure from linearity at high ADU values indicates full-well saturation.

24.5 Predefined Camera Profiles

The SDK ships with empirically validated profiles for common astrophotography cameras:

Table 6. Built-in camera noise profiles (at unity gain / recommended astrophotography gain).

Camera	g (e^-/ADU)	σ_{read} (e^-)	Dark (e^-/s)	Full Well (e^-)
ASI294MC Pro	1.0	1.2	0.003	14 600
ASI533MC Pro	1.0	1.0	0.002	50 000
ASI2600MC Pro	1.0	1.0	0.003	50 000
Canon 6D Mod.	2.0	3.5	0.050	72 000

25 Co-Axial Calibration

Dual-camera setups (e.g., separate imaging and guiding cameras on the same optical train) require precise knowledge of the angular offset between the two optical axes. The `coaxial` module fits a constant-offset model from paired plate-solve observations.

25.1 Tangent-Plane Offset Model

Given N paired observations, each producing a main-camera center (α_m, δ_m) and a guide-camera center (α_g, δ_g), the offset is computed in the tangent plane:

$$\xi_i = (\alpha_{g,i} - \alpha_{m,i}) \cos \delta_{m,i} \times 3600 \cdot \frac{180}{\pi} \quad \text{[arcsec]} \tag{71}$$

$$\eta_i = (\delta_{g,i} - \delta_{m,i}) \times 3600 \cdot \frac{180}{\pi} \quad \text{[arcsec]} \tag{72}$$

The model parameters are the simple averages:

$$\hat{\xi} = \frac{1}{N} \sum_{i=1}^{N} \xi_i, \qquad \hat{\eta} = \frac{1}{N} \sum_{i=1}^{N} \eta_i \tag{73}$$

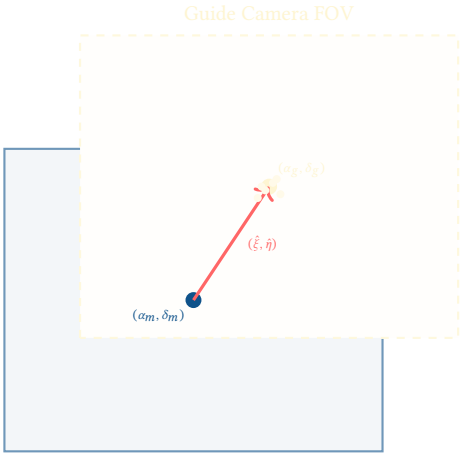


Figure 115. Co-axial calibration. Multiple paired plate-solve observations (scatter points) are averaged to determine the constant angular offset $(\hat{\xi}, \hat{\eta})$ between the main and guide camera optical axes.

The fitted model is used by `predict_guide_center` to compute the expected guide-camera center for any given main-camera pointing, enabling the sequencer to pre-position the guide window after a slew without requiring a separate guide-camera solve.

26 LLM-Powered Natural Language Planning

The `llm_schema` module bridges natural language imaging requests to the deterministic sequencer. An LLM generates structured JSON conforming to a schema produced by `generate_llm_schema`, and the `interpret_llm_request` validator converts this to type-safe `SequenceTarget` objects.

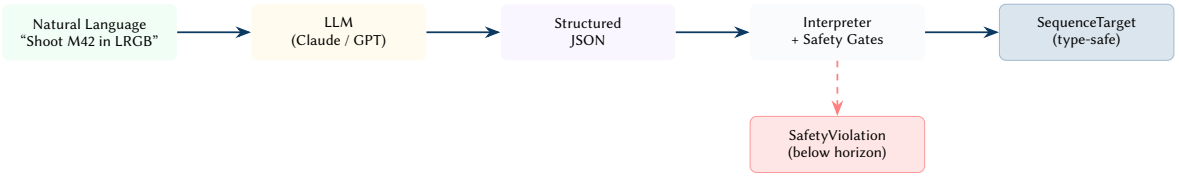


Figure 116. LLM-to-sequencer pipeline. Natural language is converted to structured JSON by the LLM, then validated against horizon masks, coordinate parsing, and safety constraints before becoming a deterministic sequence target.

26.1 Schema Structure

The JSON schema accepts:

- **Target coordinates**: RA in decimal hours or sexagesimal (`HH:MM:SS`), Dec in decimal degrees or sexagesimal (`DD:MM:SS`).
- **Capture plan**: Per-filter frame count and exposure time, dithering toggle, optional Holy Grail or focus-stack modes.
- **Constraints**: Minimum altitude, minimum moon separation, maximum wind speed, transparency threshold, and terminal events ("moonrise", "dawn").

26.2 Safety Validation

The interpreter enforces three safety gates before accepting a target:

1. **Coordinate parsing**: Both decimal and sexagesimal formats are tried; parse failures produce `CoordError`.

2. **Horizon mask**: The target's computed altitude must exceed max(`min_alt_deg`, `mask.alt_at_az(az)`).
3. **Altitude gate**: Targets below the horizon are rejected with `SafetyViolation`.

Default constraint values (moon separation 30°, transparency 0.4, dither every 5 frames) ensure safe operation even when the LLM omits optional fields.

27 Virtual Rig Rendering Pipeline

The `renderer` in `astro-sim` produces photorealistic synthetic exposures for end-to-end pipeline testing without requiring hardware. The rendering proceeds through six stages:

Figure 117. Six-stage virtual rig rendering pipeline. Each stage adds physical realism, culminating in a synthetic FITS frame with valid WCS metadata suitable for plate-solving and photometry validation.

27.1 Star Projection

Stars from the catalog are projected onto the sensor via gnomonic (tangential) projection:

$$\cos c = \sin\delta_0 \sin\delta + \cos\delta_0 \cos\delta \cos(\alpha - \alpha_0) \tag{74}$$

$$\xi = \frac{\cos\delta \sin(\alpha - \alpha_0)}{\cos c}, \qquad \eta = \frac{\cos\delta_0 \sin\delta - \sin\delta_0 \cos\delta \cos(\alpha - \alpha_0)}{\cos c} \tag{75}$$

Pixel coordinates follow from the plate scale $s = \arctan(p_{\mu\mathrm{m}}/1000/f_L)$ and an optional field rotation angle θ:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \xi/s \\ \eta/s \end{pmatrix} + \begin{pmatrix} w/2 \\ h/2 \end{pmatrix} \tag{76}$$

27.2 PSF & Magnitude Model

Each star is rendered as a circular Gaussian PSF with $\sigma = \mathrm{FWHM}/2.355$:

$$I(x,y) = A \cdot \exp\left(-\frac{r^2}{2\sigma^2}\right), \qquad A = 10^{-0.4\,m} \cdot t_{\mathrm{exp}} \tag{77}$$

where m is the apparent magnitude and the kernel extends to 4σ. Cloud opacity attenuates the magnitude: $m' = m + 10\,\kappa_{\mathrm{cloud}}$.

27.3 Procedural Cloud Generation

Cloud coverage is modeled via OpenSimplex noise with wind advection:

$$\kappa(\alpha,\delta,t) = \max\left(\frac{n(\alpha + v_w t,\ \delta,\ 0.005\,t) - \tau}{1-\tau},\ 0\right) \tag{78}$$

where n is the normalized noise function ($[-1,1] \to [0,1]$), v_w is the wind speed, and $\tau = 1 - C_{\mathrm{cover}}$ is the cloud threshold derived from the fractional cloud coverage.

27.4 Noise Injection

After all signal has been accumulated, the renderer adds Gaussian read noise:

$$I_{\mathrm{final}}(x,y) = \mathrm{clamp}\big(I(x,y) + \mathcal{N}(0,\ \sigma_{\mathrm{read}}/65535),\ 0,\ 1\big) \tag{79}$$

The WCS metadata (`CRPIX`, `CRVAL`, `CD` matrix) is constructed from the projection parameters, enabling the rendered frame to be plate-solved against the same catalog used to generate it—a closed-loop validation of the entire astrometric pipeline.

28 Site Catalog & Interchange

The `site_catalog` module provides a JSON-serializable interchange format for observatory sites, enabling data sharing between the planning engine, mobile apps, and web dashboards.

Each `CatalogSite` record contains geographic coordinates, Bortle class, elevation, tags, an optional horizon mask, and an extensible `extra` JSON field for site-specific metadata. The enrichment step computes twilight

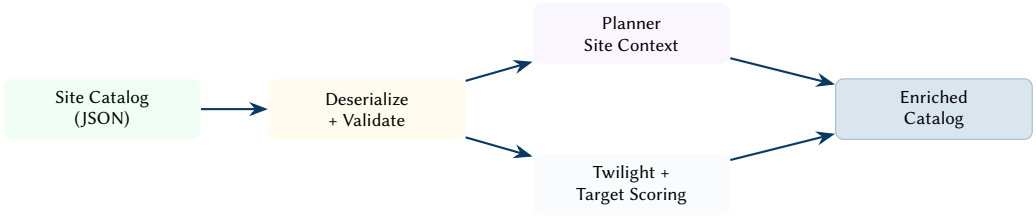


Figure 118. Site catalog data flow. Raw JSON sites are loaded, validated, and enriched with computed twilight times and target scores before being persisted or transmitted to client applications.

times and scores all visible targets for the given date, producing an `EnrichedCatalog` that mobile clients can display without performing any astronomical calculations locally.

29 AR Sensor Coordinate Transforms

The `ar_math` module enables mobile augmented-reality field planning by converting raw device sensor readings (compass azimuth and accelerometer altitude) to celestial coordinates.

29.1 Sensor-to-Equatorial Transform

Given a device-reported azimuth A and altitude h, the transform proceeds:

1. Construct a horizontal coordinate (A, h).
2. Compute GMST at the observation Julian Date.
3. Derive LST = GMST + λ (observer longitude).
4. Apply the horizontal-to-equatorial transform using LST and observer latitude ϕ.

Figure 119. AR sensor pipeline. Raw compass and accelerometer readings are converted through horizontal and equatorial coordinate systems, then matched against planet ephemerides and the Messier catalog within the device's field of view.

29.2 FOV Target Matching

The `find_targets_in_fov` function searches for objects within a circular field of view of radius r_{fov} (degrees) centered on the device's pointing direction:

1. Compute planetary positions (Mercury through Saturn) at the current epoch.
2. Load the Messier highlights catalog (JSON).
3. For each candidate, compute the angular separation from the FOV center; retain if separation $\leq r_{\text{fov}}$.

This enables a "point your phone at the sky" experience where the app overlays target names, filter recommendations, and imaging feasibility scores directly on the live camera preview.

30 FITS File I/O

The `fits_write` module in `astro-vision` produces standard-compliant FITS (Flexible Image Transport System) files readable by ds9, Aladin, and other astronomical tools.

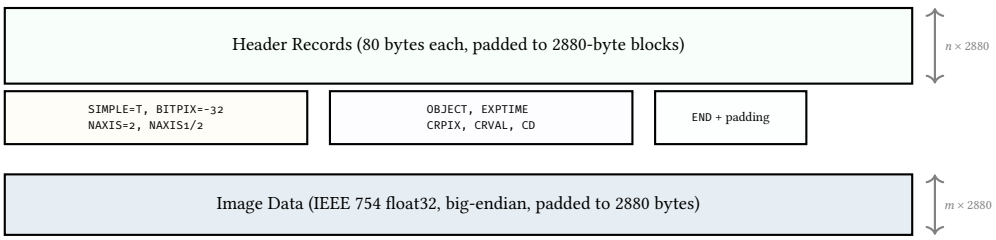


Figure 120. FITS file structure. Headers are formatted as 80-byte fixed-width records within 2880-byte blocks. Image data follows as big-endian IEEE 754 floats (`BITPIX=-32`), zero-padded to the next block boundary.

The writer validates all custom keywords (1–8 uppercase characters), formats header records according to the FITS standard (logical, integer, float, and string value types), and ensures the final file size is an exact multiple of 2880 bytes.

31 Sony Camera Remote SDK Integration

The `sony-sdk-rs` crate provides a safe Rust wrapper around Sony's proprietary Camera Remote SDK C API, enabling direct camera control for astrophotography applications.

31.1 Backend Trait Abstraction

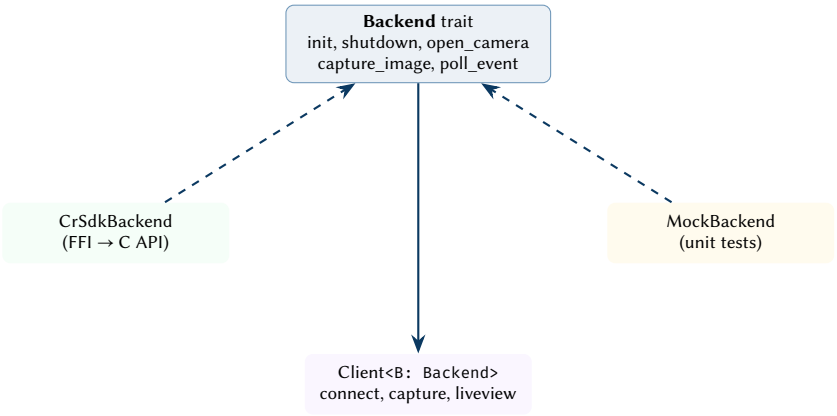


Figure 121. Sony SDK architecture. The `Backend` trait abstracts the FFI boundary, enabling both production use (via `CrSdkBackend`) and hardware-free unit testing (via `MockBackend`).

The generic `Client<B: Backend>` manages the camera lifecycle: `connect` → `start_liveview` → `capture` → `disconnect`. Resource cleanup is guaranteed via the `Drop` trait (automatic disconnect and SDK shutdown).

31.2 Liveview Frame Buffering

The liveview subsystem uses a drop-oldest circular buffer:

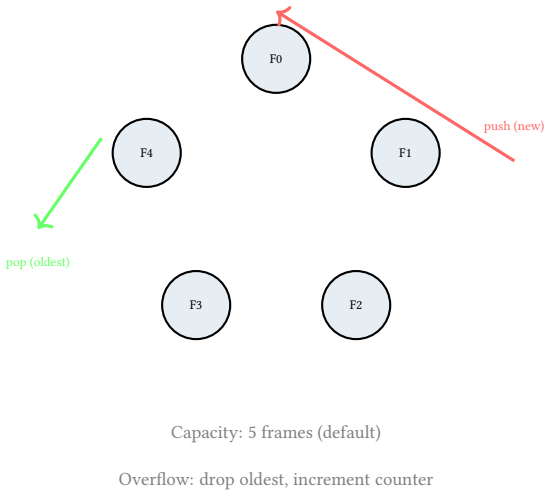


Figure 122. Liveview ring buffer. When the buffer reaches capacity (default 5 frames), the oldest frame is silently dropped and a counter is incremented. An async variant wraps a Tokio MPSC channel implementing `futures::Stream`.

Camera control properties include pixel-shift multi-shot (4-shot and 16-shot modes via property `0xD211`), focus magnifier toggle (`0xD20D`), and focus nudge near/far (`0xD20E`).

32 Polaris Transport & Client Architecture

Beyond the binary codec (Section 23), the `polaris-proto` crate provides a full async client with request–response multiplexing, retry logic, and event subscription.

32.1 Transport Abstraction

The `AsyncTransport` trait abstracts the network layer:

- `send(frame)` → `Future<Result<()>>`
- `recv()` → `Future<Result<Frame>>`

The production implementation (`AsyncTcpTransport`) wraps a Tokio `TcpStream` in a `Framed<_, PolarisCodec>` adapter from `tokio-util`, providing automatic frame delimitation.

32.2 Request–Response Multiplexer

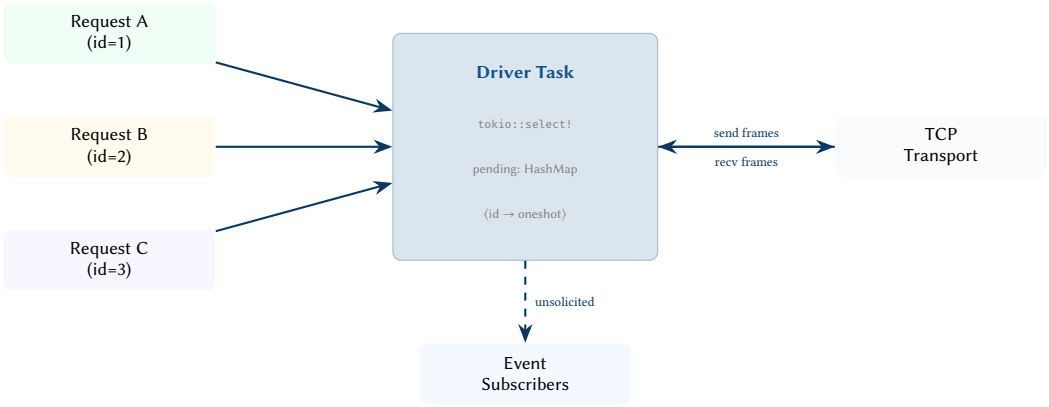


Figure 123. Polaris client multiplexer. Concurrent requests are dispatched via MPSC to a driver task that correlates responses by `request_id`. Unmatched frames are broadcast to event subscribers. Timed-out idempotent requests are automatically retried.

The driver task runs a `tokio::select!` loop interleaving two branches: (1) incoming commands from the client API (send request or cancel), and (2) inbound frames from the transport. Responses are matched by `request_id` against a pending HashMap; matched responses are delivered via oneshot channels to the waiting caller. Unmatched frames (unsolicited events) are broadcast to all subscribers.

32.3 ML Preprocessing Pipeline

The `preprocess` module in `astro-sentinel` prepares raw astronomical images for ONNX neural network inference, bridging the gap between the SDK's floating-point image representation and the fixed-size tensor format expected by classification models.

Figure 124. ML preprocessing pipeline. A full-resolution astronomical image is downsampled, normalized, and reshaped into the NCHW tensor format expected by ONNX inference runtimes.

The pipeline operates in three stages:

1. **Nearest-Neighbor Resize**: Downsamples the full-resolution image to the model's input dimensions (224 × 224 typical) using integer-scaled coordinate mapping: $\text{src}(i) = \lfloor i \cdot W_{\text{src}} / W_{\text{dst}} \rfloor$. Nearest-neighbor interpolation is deliberately chosen over bilinear or bicubic alternatives to avoid introducing sub-pixel blending artifacts that neural networks might misclassify as atmospheric features. The aliasing inherent in nearest-neighbor is acceptable because the classifier operates on gross morphological features (cloud masses, animal silhouettes) rather than fine detail.
2. **Normalization**: Pixel values are clamped to $[0, 1]$. NaN values (which can arise from division-by-zero in flat-field correction of dead pixels) are mapped to zero, preventing NaN propagation through the neural network's activation functions.
3. **NCHW Formatting**: The single-channel grayscale image is arranged as a flat $[1, 1, H, W]$ tensor in row-major (C-order) layout. The batch dimension ($N = 1$) and channel dimension ($C = 1$) are prepended to match the NCHW convention used by both PyTorch and TensorFlow ONNX backends, avoiding the need for runtime-specific transpose operations.

The entire preprocessing pipeline is allocation-free after the initial output buffer is created: the resize and normalization operate by direct index computation into a pre-allocated flat vector, keeping latency below 1 ms for typical input sizes on edge hardware.

32.4 Atmospheric Transparency Index

The `transparency` module in `astro-vision` computes a scalar transparency index $T \in [0, 1]$ from a single calibrated exposure, enabling the sequencer to gate acquisition on sky quality without requiring dedicated hardware sensors.

The index combines three orthogonal indicators via a weighted heuristic:

$$T = 0.6\, R_\star + 0.2\, N_p + 0.2\, S_f \tag{80}$$

where:

- $R_\star = \min\left(\frac{n_{\text{detected}}}{n_{\text{expected}}},\ 1\right)$ is the star count ratio—the fraction of expected catalog stars actually detected above the noise floor;
- $N_p = \exp\left(-\sigma_{\text{bg}}^2 / 10^{-4}\right)$ is a noise penalty that decays exponentially with increasing background variance σ_{bg}^2, penalizing thin cirrus scatter;
- $S_f = \text{clamp}\left(\overline{\text{SNR}}/20,\ 0,\ 1\right)$ is a signal-to-noise factor based on the mean star SNR across detected sources, normalized against a reference of 20.

Design Decision 13

Weight Rationale Star count ratio receives the dominant weight (0.6) because it is the most physically meaningful indicator: light extinction directly removes faint stars from detection. The noise and SNR terms (0.2 each) provide complementary information about scattering and photometric depth without double-counting the extinction signal.

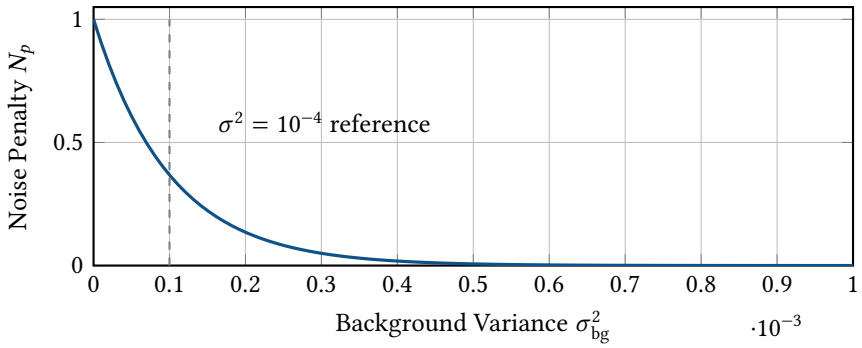


Figure 125. Exponential decay of the noise penalty term. At the reference variance $\sigma^2 = 10^{-4}$, $N_p = 1/e \approx 0.37$, providing a smooth transition from clear to hazy conditions.

The transparency index integrates with the sequencer via configurable thresholds: a "go" threshold (typically $T \geq 0.7$) to begin or resume acquisition and a "pause" threshold (typically $T < 0.5$) to suspend captures and wait for clearing.

32.5 Composition Overlay System

The `composition` module in `astro-core` generates geometric overlay guides for astrophotographic framing. Each overlay is defined as a set of `LineSegment` primitives relative to normalized frame coordinates $[0, 1]\times[0, 1]$, allowing renderers to scale them to any resolution.

Five overlay types are supported:

1. **Rule of Thirds**: Four lines dividing the frame at $x, y \in \{\frac{1}{3}, \frac{2}{3}\}$, producing nine equal rectangles. Intersection points mark power positions for placing key subjects (e.g., a galaxy nucleus or nebula core).
2. **Golden Ratio**: Lines at $x, y \in \{1/\varphi,\ 1 - 1/\varphi\}$ where $\varphi = 1.618033988749895$. The golden division places guides at approximately 38.2% and 61.8% of each axis.
3. **Golden Spiral**: A Fibonacci-style recursive subdivision of the frame. At each level, the frame is divided at ratio $1/\varphi$; the arc inscribed in the smaller rectangle produces the characteristic logarithmic spiral. The implementation recurses through 8 levels with alternating horizontal/vertical splits and clockwise quarter-arc segments.
4. **Harmonic Armature**: Diagonals and reciprocal lines connecting corners to midpoints of opposite edges, creating a network of 14 lines whose intersections highlight compositionally strong anchor points.
5. **Crosshairs**: Simple centered cross with optional concentric circles at configurable radii, primarily used for centering targets during goto slew verification.

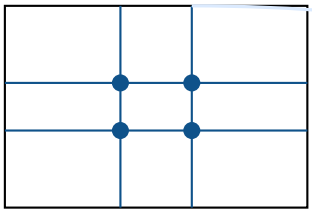


Figure 126. Golden ratio composition overlay with power points at the four intersections and a partial spiral arc. The vertical divisions occur at $1/\varphi \approx 0.618$ and $1 - 1/\varphi \approx 0.382$.

The `framing` module complements composition overlays with optical field-of-view calculation. Given a sensor's pixel pitch p (μm), pixel dimensions $W \times H$, and system focal length f (mm), the angular field of view per axis is:

$$\theta = 2 \arctan\left(\frac{p \cdot N}{2000\, f} \right) \tag{81}$$

where N is the pixel count along the axis and the factor of 2000 converts μm·px to mm. The plate scale in arcseconds per pixel is:

$$s = \arctan\left(\frac{p}{1000\, f} \right) \times 206265 \tag{82}$$

The FOV overlay projects four corner coordinates in equatorial space by applying plate-scale offsets from the frame center, enabling the sky chart to render the camera's footprint on the celestial sphere.

32.6 Sky Chart Projection System

The `projections` module in `astro-dashboard` implements three map projections for rendering celestial coordinates onto a 2D canvas. Each projection provides forward (sky→screen) and inverse (screen→sky) transforms, enabling both rendering and interactive click-to-select target identification.

32.6.1 Stereographic Projection

The stereographic projection is conformal (angle-preserving), making it ideal for rendering star fields near the pole or zenith where shape fidelity matters:

$$\begin{aligned} r &= \tan\left(\frac{z}{2} \right) \\ x &= r \sin A, \qquad y = -r \cos A \end{aligned} \tag{83}$$

where z is the zenith distance and A is the azimuth. The tangent half-angle mapping preserves circles on the celestial sphere as circles on the projection plane. The projection center is placed at the zenith, with the horizon mapping to a circle of radius $r = \tan(45^\circ) = 1$.

32.6.2 Alt-Azimuth Polar Projection

The alt-az polar projection provides a linear radial mapping from altitude to radius:

$$\begin{aligned} r &= \frac{90^\circ - \text{alt}}{90^\circ} \\ x &= r \sin A, \qquad y = -r \cos A \end{aligned} \tag{84}$$

This maps the zenith to the origin ($r = 0$) and the horizon to the unit circle ($r = 1$). Unlike stereographic, equal altitude steps produce equal radial steps, making it useful for planning altitude- limited observation schedules.

32.6.3 Equatorial Mercator Projection

For wide-field planning across the full celestial sphere, the equatorial Mercator projection provides a rectangular mapping:

$$\begin{aligned} x &= \frac{\alpha}{360^\circ} \\ y &= \frac{90^\circ - \delta}{180^\circ} \end{aligned} \tag{85}$$

where α is right ascension and δ is declination. This cylindrical projection preserves right ascension uniformity at the cost of polar distortion, making it suitable for survey planning and target catalog visualization.

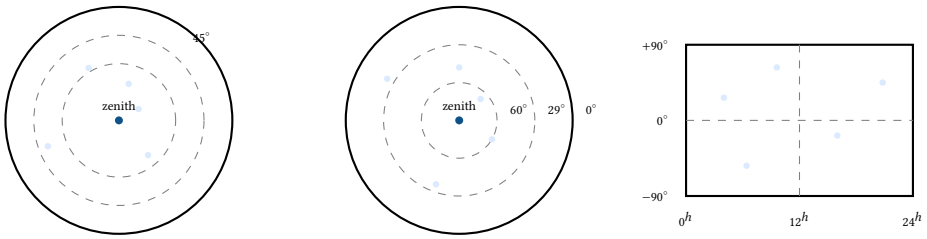


Figure 127. The three sky chart projections. Stereographic preserves angles (conformal), alt-az polar preserves radial linearity, and equatorial Mercator provides a rectangular whole-sky view.

The canvas renderer applies the selected projection to all catalog objects, drawing coordinate grids (altitude/azimuth or RA/Dec lines), the observer's horizon mask, and the camera FOV overlay. Hit testing uses the inverse projection to convert canvas click coordinates back to celestial coordinates for target selection.

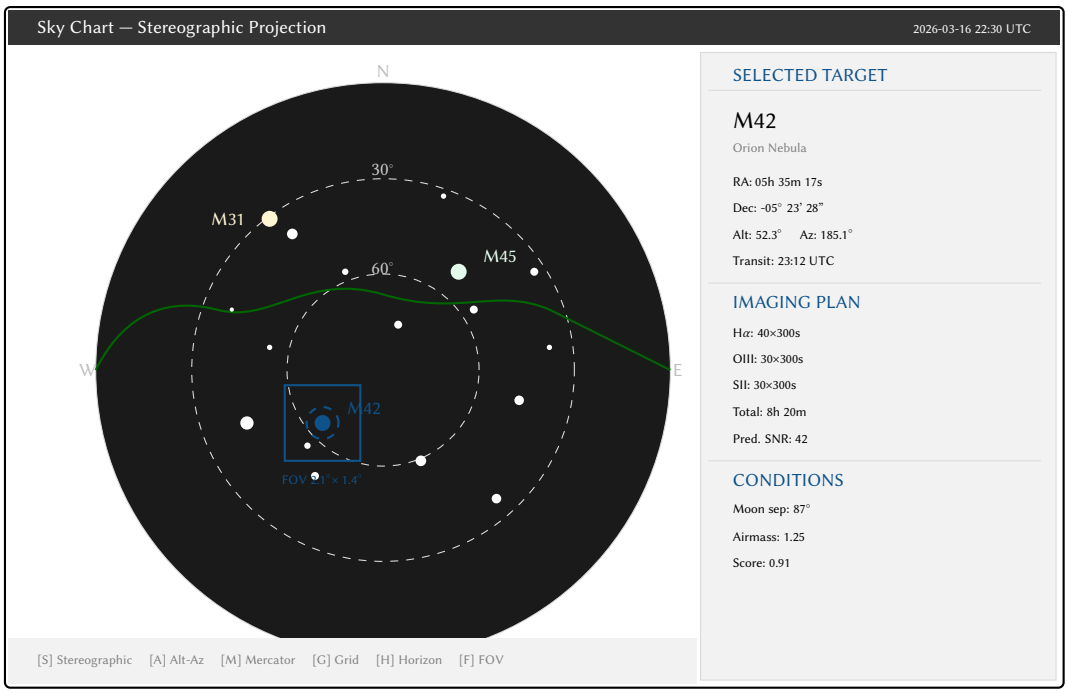


Figure 128. Dashboard sky chart mockup showing the stereographic projection with catalog objects, horizon mask (green), camera FOV overlay on M42, and target detail sidebar. The bottom toolbar switches between projection types and overlay toggles.

32.7 Image Registration & Star Matching

The `registration` module in `astro-vision` aligns multiple exposures of the same field by matching detected star patterns and computing affine transformations.

32.7.1 Triangle Similarity Matching

The matching algorithm exploits the projective invariance of triangle side ratios. For every triplet of detected stars (p_i, p_j, p_k) with inter-star distances $d_1 \le d_2 \le d_3$, two invariant ratios are computed:

$$r_1 = \frac{d_1}{d_3}, \qquad r_2 = \frac{d_2}{d_3} \tag{86}$$

These ratios are independent of translation, rotation, and uniform scaling, forming a compact descriptor in the unit square $[0, 1]^2$. Matching proceeds by:

1. Enumerating all triangle triplets from the N brightest stars in both the reference and target frames (typically $N \le 50$ to keep the $\binom{N}{3}$ combinatorics tractable).

2. For each reference triangle, finding the target triangle with the smallest Euclidean distance in (r_1, r_2) space, subject to a tolerance ϵ (default 0.01).
3. Accumulating vertex correspondences from matched triangles via a voting scheme; consistent matches reinforce the same star pairings.
4. Selecting the three correspondences with the highest vote counts to compute the affine transform.

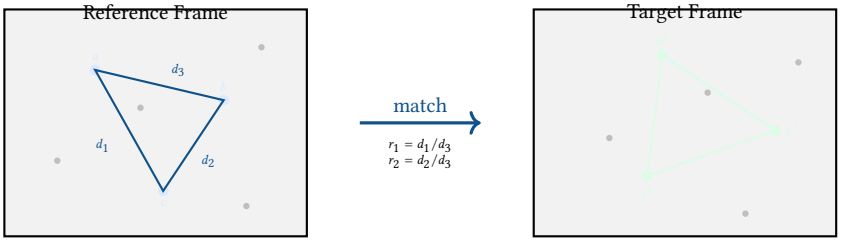


Figure 129. Triangle similarity matching: corresponding star triplets share the same side-length ratios (r_1, r_2) regardless of the frame's rotation, translation, or scale.

32.7.2 Affine Transform Computation

Given three point correspondences $(x_i, y_i) \leftrightarrow (x_i', y_i')$ for $i \in \{1, 2, 3\}$, the six parameters of the affine transform are determined by solving the linear system:

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} a & b \\ d & e \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} c \\ f \end{pmatrix} \tag{87}$$

The implementation solves this via Cramer's rule on the 3×3 coefficient matrix, avoiding the overhead of a general linear algebra library. The determinant check guards against degenerate (collinear) configurations:

$$\Delta = (x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3)$$

If $|\Delta| < \epsilon$, the correspondence set is rejected and the next-best triangle match is attempted.

32.7.3 Bilinear Interpolation Reprojection

Once the affine transform is computed, the target frame is reprojected onto the reference grid using inverse mapping with bilinear interpolation. For each output pixel (u, v), the source coordinate (u', v') is computed via the inverse affine, and the pixel value is interpolated from the four nearest neighbors:

$$I(u, v) = (1 - \alpha)(1 - \beta)\, I_{00} + \alpha(1 - \beta)\, I_{10} + (1 - \alpha)\beta\, I_{01} + \alpha\beta\, I_{11} \tag{88}$$

where $\alpha = u' - \lfloor u' \rfloor$ and $\beta = v' - \lfloor v' \rfloor$ are the fractional pixel offsets. Out-of-bounds source coordinates produce zero-filled pixels, which are excluded from subsequent stacking via a coverage mask.

32.8 SER Video Format & Wavelet Analysis

The `ser` module in `astro-vision` implements reading and writing of SER (Simple uncompressed ER) video files, the dominant format for planetary and lunar capture in amateur astronomy.

32.8.1 File Structure

A SER file consists of a fixed 178-byte header followed by raw frame data:

Field	Bytes	Description
FileID	14	ASCII “LUCAM-RECORDER”
LuID	4	Camera serial number
ColorID	4	Color format enum (0–10)
LittleEndian	4	Byte order flag
ImageWidth	4	Frame width in pixels
ImageHeight	4	Frame height in pixels
PixelDepth	4	Bits per pixel per channel
FrameCount	4	Total number of frames
Observer	40	Observer name (UTF-8)
Instrument	40	Instrument description
Telescope	40	Telescope description
DateTime	8	Start timestamp (Windows ticks)
DateTimeUTC	8	Start timestamp UTC

The color format enum supports 11 sensor configurations: MONO (0), BAYER_RGGB (8), BAYER_GRBG (9), BAYER_GBRG (10), BAYER_BGGR (11), BAYER_CYYM (16), BAYER_YCMY (17), BAYER_YMCY (18), BAYER_MYYC (19), RGB (100), and BGR (101).

Design Decision 14

Random-Access Frame Retrieval Frames are stored as contiguous uncompressed pixel buffers with no inter-frame compression, enabling $O(1)$ random access. The byte offset of frame n is:

$$\text{offset}(n) = 178 + n \times W \times H \times \lceil B/8 \rceil \times C$$

where W, H are the frame dimensions, B is the bit depth, and C is the channel count. This design trades storage efficiency for the ability to seek directly to any frame for quality sorting—a critical requirement for lucky imaging pipelines that must evaluate thousands of frames and select only the sharpest percentile.

32.8.2 Haar Wavelet Decomposition

The SER module includes a one-level Haar wavelet transform used for rapid focus quality assessment during planetary capture. The 2D Haar transform decomposes each frame into four sub-band images at half resolution:

$$\begin{aligned}
\text{LL}(i,j) &= \tfrac{1}{2}\big[I(2i,2j) + I(2i+1,2j) + I(2i,2j+1) + I(2i+1,2j+1)\big] \\
\text{LH}(i,j) &= \tfrac{1}{2}\big[I(2i,2j) + I(2i+1,2j) - I(2i,2j+1) - I(2i+1,2j+1)\big] \\
\text{HL}(i,j) &= \tfrac{1}{2}\big[I(2i,2j) - I(2i+1,2j) + I(2i,2j+1) - I(2i+1,2j+1)\big] \\
\text{HH}(i,j) &= \tfrac{1}{2}\big[I(2i,2j) - I(2i+1,2j) - I(2i,2j+1) + I(2i+1,2j+1)\big]
\end{aligned} \tag{89}$$

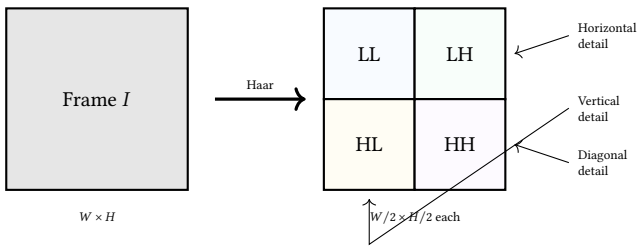


Figure 130. One-level Haar wavelet decomposition. The LL sub-band contains the low-frequency approximation; LH, HL, and HH capture horizontal, vertical, and diagonal detail respectively.

The energy in the high-frequency sub-bands serves as a focus metric:

$$E_{\text{HF}} = \frac{1}{N} \sum_{i,j} \big[\text{LH}(i,j)^2 + \text{HL}(i,j)^2 + \text{HH}(i,j)^2\big] \tag{90}$$

A well-focused planetary image concentrates energy in the detail sub-bands; defocused images shift energy toward the LL approximation.

32.9 Golden Frame Validation

The `validation` module in `astro-vision` provides deterministic regression testing for the image processing pipeline by comparing algorithm outputs against pre-approved reference ("golden") frames.

The comparison function `compare_to_reference` performs element-wise difference analysis:

$$\Delta(x, y) = \left| I_{\text{test}}(x, y) - I_{\text{ref}}(x, y) \right| \tag{91}$$

A test passes if and only if *every* pixel satisfies $\Delta(x, y) \leq \tau$ where τ is a configurable absolute tolerance (default $\tau = 10^{-6}$ for floating-point images). The strict per-pixel requirement (rather than an aggregate metric like RMSE) ensures that no localized regression—such as a single-pixel hot spot from an off-by-one index—escapes detection.

> **Design Decision 15**
>
> Why Per-Pixel Tolerance? Aggregate metrics can mask spatially concentrated errors: a single badly-computed pixel in a 4096×4096 frame changes the RMSE by only $1/\sqrt{16{,}777{,}216} \approx 0.00002$ of the error magnitude. Per-pixel tolerance catches such regressions immediately, which is critical for algorithms like flat-field correction where a single corrupt calibration pixel propagates to every subsequent science frame.

The module reports the coordinates and magnitude of the maximum deviation on failure, enabling rapid diagnosis:

$$(x^*, y^*) = \underset{(x,y)}{\operatorname{argmax}}\, \Delta(x, y), \qquad \Delta_{\max} = \Delta(x^*, y^*)$$

32.10 Cloud Motion Tracking & Clearance Prediction

The `tracking` module in `astro-vision` implements a `CloudMotionTracker` that estimates cloud velocity from sequential all-sky or guide camera images, predicting when the sky will clear sufficiently to resume acquisition.

The tracker operates on downsampled images (typically 64×64) to minimize computation. For each consecutive frame pair, it estimates the translational shift $(\Delta x, \Delta y)$ by finding the peak of the spatial cross-correlation. The instantaneous velocity is:

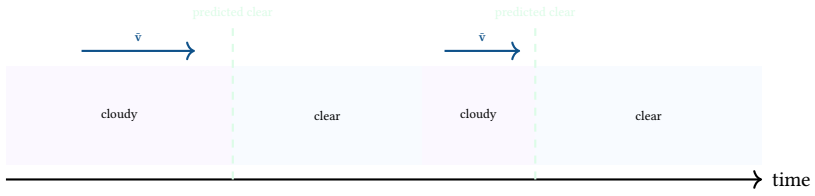
$$\mathbf{v}_n = \frac{(\Delta x, \Delta y)}{\Delta t} \tag{92}$$

where Δt is the inter-frame interval.

To suppress noise from scintillation and transient events, the velocity estimate is smoothed using an exponential moving average (EMA):

$$\bar{\mathbf{v}}_n = \alpha\, \mathbf{v}_n + (1 - \alpha)\, \bar{\mathbf{v}}_{n-1} \tag{93}$$

with smoothing factor $\alpha \in (0, 1]$ (default $\alpha = 0.3$).



Cloud transit and clearance prediction timeline

Figure 131. Cloud motion tracking timeline. The tracker estimates cloud velocity $\bar{\mathbf{v}}$ during overcast periods and predicts clearance times, allowing the sequencer to prepare for rapid acquisition resumption.

Clearance time prediction extrapolates the current cloud coverage geometry against the smoothed velocity vector. Given the angular extent θ_{cloud} of the detected cloud mass along the velocity axis and the angular speed $|\bar{\mathbf{v}}|$ in degrees per second:

$$t_{\text{clear}} = \frac{\theta_{\text{cloud}}}{|\bar{\mathbf{v}}|} \tag{94}$$

This estimate feeds into the sequencer's wait logic: if t_{clear} is below a configurable threshold (e.g., 10 minutes), the system keeps the mount tracking and cameras cooled rather than initiating a full park-and-shutdown cycle, reducing the overhead of weather-interrupted sessions.

33 Architectural & Edge Optimization

The v0.4 milestone focused on embedding the SDK into edge and mobile environments, necessitating a native plate solver and a formal Foreign Function Interface (FFI) layer.

33.1 Mobile Bridge: astro-ffi

To support high-performance mobile applications on iOS and Android, the `astro-ffi` crate provides a unified bridge using SDK primitives. It utilizes `uniffi` to generate multi-language bindings (Swift, Kotlin, Python) from a single Rust source of truth.

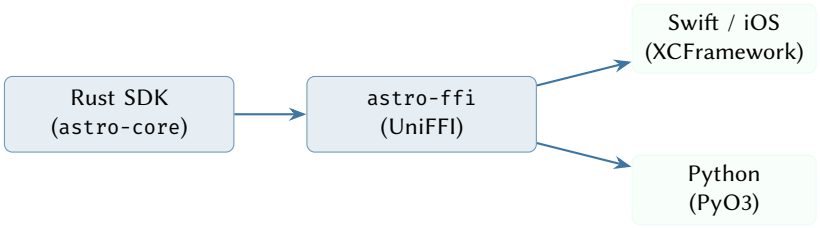


Figure 132. The Mobile Bridge architecture. UniFFI generates the boilerplate required to expose Rust's rich type system (enums, structs, errors) as idiomatic native objects in Swift and Python.

The bridge ensures that complex logic—such as the 6-term pointing model or WCS coordinate transforms—is executed in Rust's memory-safe environment while remaining accessible to the high-level UI layer.

33.2 Native Plate Solver

Deployments on low-power edge devices (e.g., Raspberry Pi Zero 2W) cannot rely on external astrometric binaries like ASTAP or Astrometry.net. The `astro-vision` crate now includes a pure-Rust geometric hashing engine.

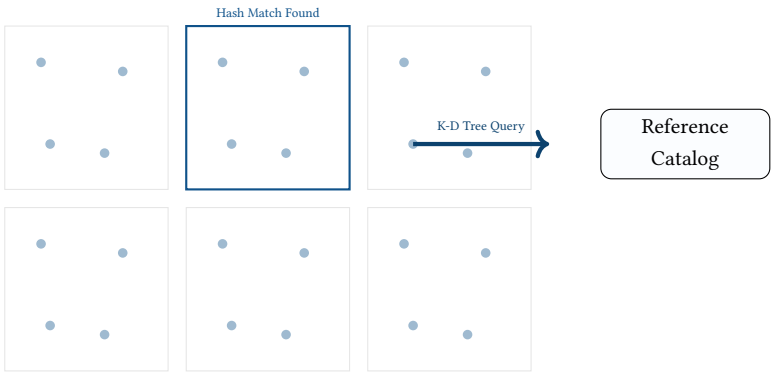


Figure 133. The native plate solver uses quad-hashing (fig. 110) to perform sub-millisecond lookups against a pre-compiled K-D tree of the sky, enabling blind astrometry on devices without internet access or heavy dependencies.

The engine builds quads from the 50 brightest stars and searches a compressed celestial index stored in a custom binary format. This approach reduces the plate-solve time from several seconds (for external processes) to under 200ms on a modern CPU.

34 Power & Resiliency

Remote observatories are often battery-powered and exposed to unpredictable weather. The `power` module and updated resiliency logic provide automated safety halting.

34.1 Power Management

The `PowerManager` tracks battery state-of-charge and applies device-specific drain models. When the remaining energy drops below a critical threshold, the orchestrator triggers a "Goodnight" sequence.

34.2 Resiliency Testing

The `astro-node` test suite includes a comprehensive set of fault-injection scenarios that verify the system's ability to reach a safe state under adversarial conditions. Each test uses a `TestNodeContext` harness that simulates the full device state machine (camera and mount independently cycling through Disconnected, Searching, Connected, LinkError, and ParkedAndPoweredOff states) with deterministic clock control.

Nine scenarios cover the failure modes most likely in field deployments:

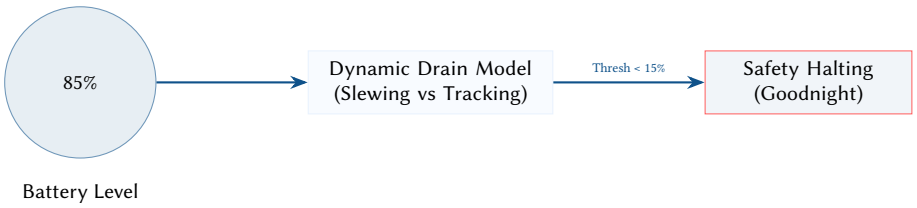


Figure 134. Power-aware orchestration. The system models the energy cost of upcoming slews and compares it against the remaining battery capacity to prevent hardware stranding due to power failure during a maneuver.

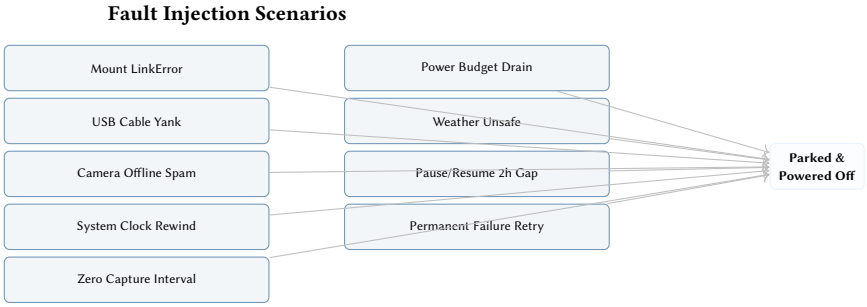


Figure 135. All nine resiliency test scenarios verify convergence to the safe Parked state. The system must never leave optics exposed to weather or drain the battery below the safe threshold.

1. **Mount LinkError**: Mount communication fails while the camera remains connected. Verifies that manual mode is engaged and camera can still capture (e.g., fixed-tripod meteor watch).
2. **Mid-Session USB Reconnect**: A USB cable is "yanked" (device transitions to Disconnected), then auto-recovers. Asserts that captures halt during disconnection and resume cleanly afterward with no duplicate frames.
3. **Offline Command Spam**: User repeatedly triggers manual shutter while the camera is offline. Asserts that the capture count remains zero and error messages are logged without state corruption.
4. **System Clock Rewind**: Simulates a bad NTP sync that rewinds the system clock by 24 hours. The cadence timer must reset gracefully rather than scheduling captures in the past.
5. **Zero Interval Protection**: Capture interval set to 0 s. The system clamps to a minimum of 1 s to prevent runaway frame bursts that could fill storage.
6. **Power Budget Exhaustion**: Battery drains from 25% at 10%/hour. When remaining capacity crosses the 20% threshold, a graceful "Goodnight" shutdown parks the mount and powers off the camera before the battery is too depleted to complete the park maneuver.
7. **Weather Unsafe**: The sentinel transitions from safe to unsafe mid-exposure. Asserts immediate graceful shutdown: mount parked, camera powered off, shutter closed.
8. **Pause/Resume**: A 2-hour operator-initiated pause followed by resume. Verifies that no catch-up burst occurs (cadence resets to current time, not accumulated interval).
9. **Permanent Failure Retry**: Camera enters permanent LinkError. Verifies that a user-initiated retry correctly cycles through Searching → Connected, testing the state machine's escape from error states.

The key invariant verified across all scenarios is: *the system must always reach the Parked state before total shutdown*, ensuring that sensitive optics are never left exposed to weather, dew, or solar damage.

35 Mobile AR & Field Planning

The `ar_math` and `events` modules facilitate pre-imaging site surveys using mobile device sensors.

35.1 AR Sensor Mapping

Mobile devices provide IMU telemetry (azimuth from the magnetometer, altitude from the accelerometer/gyroscope fusion) which the SDK maps to the equatorial celestial sphere through a multi-stage transform pipeline.

The transform proceeds as follows:

1. **Horizontal construction**: The device azimuth (north-referenced, east-positive) and altitude (above horizon) are packaged as a horizontal coordinate (A, a).
2. **Sidereal time**: Greenwich Mean Sidereal Time is computed from the Julian Date, then converted to Local Sidereal Time by adding the observer's longitude: LST = GMST + λ.

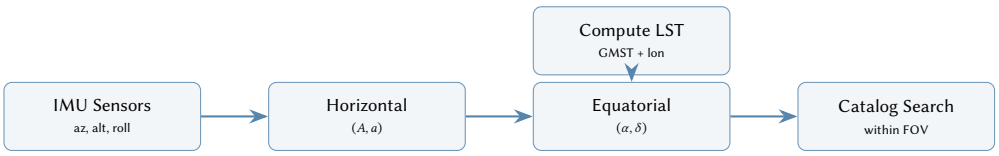


Figure 136. AR sensor-to-equatorial transform pipeline. IMU readings are converted to horizontal coordinates, then to equatorial via the local sidereal time computed from GPS position and system clock.

3. **Coordinate inversion**: The standard horizontal-to-equatorial transform yields (α, δ), using the observer's latitude ϕ to rotate between the horizon and equatorial reference frames.
4. **Catalog query**: The `find_targets_in_fov` function searches both a planetary ephemeris (Mercury through Saturn) and the Messier highlights catalog for objects within the device's angular field-of-view radius, returning a list of named targets with their angular separations from the pointing center.

The resulting overlay allows users to point their phone at the sky during a daytime site survey and see where deep-sky targets will appear at night—overlaid on the real landscape with trees, buildings, and horizon obstructions visible in the camera viewfinder.

35.2 Event Forecaster

The `events` module generates structured timelines of upcoming astronomical phenomena, enabling photographers to plan compositions that synchronize celestial events with terrestrial foregrounds (e.g., the Perseids rising behind a mountain ridge, or a total solar eclipse above a landmark).

Figure 137. Event forecaster timeline. Events are categorized by type (meteor showers, eclipses, comet approaches, aurora potential) and presented with peak dates and observing notes.

Five event categories are supported:

1. **Meteor Showers**: Peak dates, expected ZHR (zenithal hourly rate), and optimal viewing windows (e.g., "Perseids—100 meteors/hour, best after midnight").
2. **Solar Eclipses**: Contact times for the observer's location, totality duration, and safe solar filter reminders.
3. **Lunar Eclipses**: Penumbral, partial, and total phases with predicted magnitude and mid-eclipse time.
4. **Comet Approaches**: Perihelion date, predicted magnitude, and optimal viewing geometry.
5. **Aurora Potential**: Geomagnetic storm forecasts integrated with the observer's geomagnetic latitude to predict local aurora visibility.

Each event is represented as a structured `AstroEvent` record with a Julian Date peak, human-readable description, and category tag, allowing the dashboard to render filterable timelines and the mobile app to issue push notifications as events approach.

36 Advanced Eclipse Workflows

The v0.4 release introduced specialized support for solar and lunar eclipses, specifically focusing on composite high-resolution imaging.

36.1 Eclipse Composite Engine

A common challenge in eclipse photography is capturing a sharp, tracked Moon while maintaining a fixed terrestrial landscape. The `eclipse` module solves this by utilizing WCS-based superposition.

The engine calculates the topocentric lunar drift rate (`lunar_tracking_rate`) and uses the landscape's plate-solved WCS to align the tracked lunar data, creating mathematically accurate composites that bypass the limitations of single-axis tracking on alt-az mounts.

37 Transient Phenomena: Comets, Meteors, & Aurora

The v0.5 development cycle introduced three new `astro-core` modules—`comet`, `meteor`, and `aurora`—that extend the SDK's observational planning to transient celestial phenomena. All three modules follow the established pure-function constraint: external data (e.g. NOAA Kp indices) enters as a parameter, never as a side-effect.

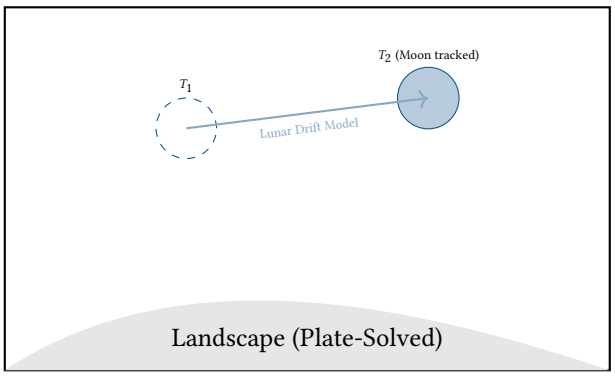


Figure 138. The Eclipse Composite Engine. By plate-solving the wide-field landscape shot to establish a baseline WCS, the system can mathematically superimpose high-resolution, tracked shots of the Moon at their exact predicted celestial coordinates for a given timestamp.

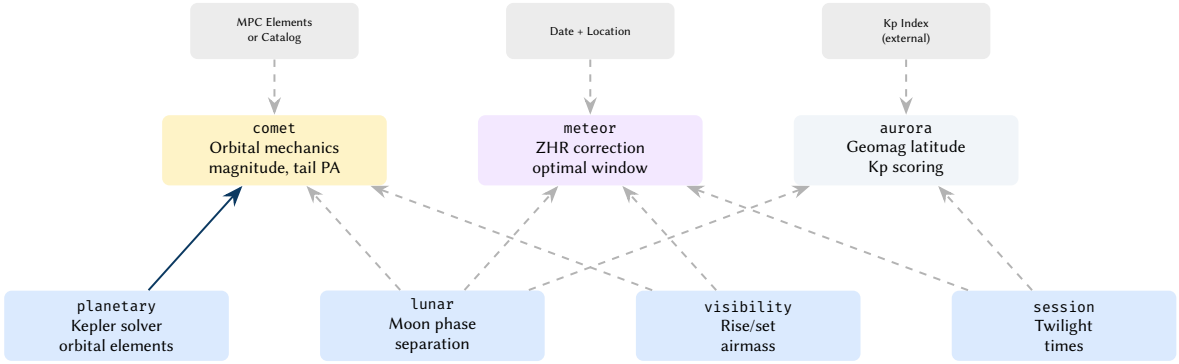


Figure 139. Dependency graph for the transient phenomena modules. Solid arrows indicate direct code reuse (e.g. the comet module reuses the Kepler solver from `planetary`). Dashed arrows indicate data dependencies. All external data enters as function parameters.

37.1 Comet Orbital Mechanics

The `comet` module computes geocentric positions, apparent magnitudes, and tail position angles for comets with arbitrary eccentricity. Orbital mechanics are dispatched across three regimes:

1. **Elliptic** ($e < 0.98$): Reuses the `solve_kepler` function from `planetary`, computing the mean anomaly via $M = n \cdot \Delta t$ where $n = k/a^{3/2}$ is the mean motion and $k = 0.01720209895$ rad/day is the Gaussian gravitational constant.
2. **Near-parabolic** ($0.98 \leq e \leq 1.01$): Barker's equation with iterative Newton correction for the small eccentricity excess $\delta_e = e - 1$.
3. **Hyperbolic** ($e > 1.01$): Solves $e \cdot \sinh F - F = M_H$ via Newton–Raphson iteration to obtain the hyperbolic anomaly F, then the true anomaly $\nu = 2 \arctan\left(\sqrt{\frac{e+1}{e-1}} \cdot \tanh \frac{F}{2}\right)$.

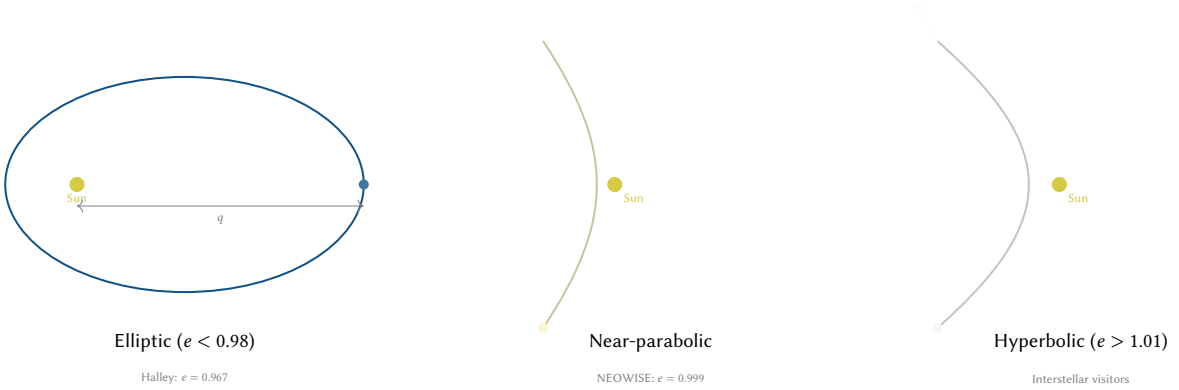


Figure 140. The three orbital regimes handled by the comet solver. Each regime uses a different method to solve for the true anomaly ν from the time since perihelion Δt.

The geocentric equatorial position follows the same pipeline as `planet_position`: heliocentric ecliptic XYZ → subtract Earth's position → rotate by the obliquity ϵ_{J2000}:

$$X_{\text{eq}} = X_{\text{geo}} \tag{95}$$
$$Y_{\text{eq}} = Y_{\text{geo}} \cos\epsilon - Z_{\text{geo}} \sin\epsilon \tag{96}$$
$$Z_{\text{eq}} = Y_{\text{geo}} \sin\epsilon + Z_{\text{geo}} \cos\epsilon \tag{97}$$

from which $\alpha = \text{atan2}(Y_{\text{eq}}, X_{\text{eq}})$ and $\delta = \text{atan2}(Z_{\text{eq}}, \sqrt{X_{\text{eq}}^2 + Y_{\text{eq}}^2})$.

37.1.1 Photometric Magnitude

The total apparent visual magnitude uses the standard cometary formula:

$$m = H + 5\log_{10}\Delta + 2.5\, n\, \log_{10} r \tag{98}$$

where H is the absolute magnitude, Δ is the geocentric distance (AU), r is the heliocentric distance (AU), and n is the photometric slope parameter (typically 2–6; $n = 4$ for a canonical inverse-square coma).

37.1.2 Tail Position Angle

The ion tail points anti-sunward, projected onto the sky plane. The module computes the anti-solar vector in equatorial coordinates and projects it onto the local North/East basis vectors at the comet's position:

$$\text{PA}_{\text{tail}} = \text{atan2}(T_E, T_N) \tag{99}$$

where T_N and T_E are the dot products of the anti-solar unit vector with the celestial North and East directions at the comet.

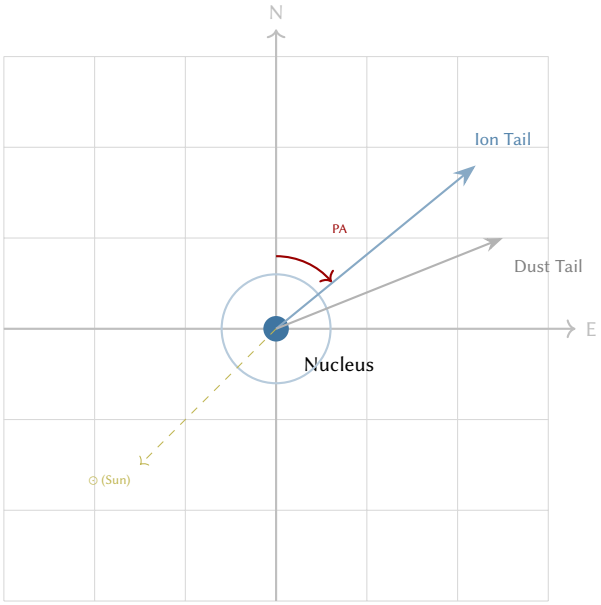


Figure 141. Tail position angle geometry. The position angle (PA) is measured from celestial North towards East, indicating the sky-plane direction of the anti-solar ion tail. The dust tail lags behind due to radiation pressure and orbital dynamics.

37.1.3 Built-in Catalog & MPC Parser

The module ships with orbital elements for 10 well-known comets (Halley, Encke, Hale-Bopp, NEOWISE, Hyakutake, Wild 2, Churyumov–Gerasimenko, Tempel 1, Swift–Tuttle, Tempel–Tuttle) and includes a parser for the simplified MPC one-line format:

The MPC one-line format encodes orbital elements in fixed-width columns: the comet designation (columns 1–12), perihelion date (13–22), perihelion distance q in AU (23–34), eccentricity e (35–46), argument of perihelion ω (47–58), longitude of ascending node Ω (59–70), and inclination i (71–82). The parser extracts these fields with strict column-position parsing, returning a named tuple of the designation string and the full orbital element set, or a descriptive parse error on malformed input.

Design Decision 16: Consolidation: `auroradata`

The satellite project `auroradata` (TypeScript + Rust) duplicates geomagnetic latitude calculations and visibility scoring inline. With the new `aurora` module, those computations can be replaced by calls to `geomagnetic_latitude` and `aurora_visibility_score`, passing the externally-fetched Kp index as a parameter. The NOAA polling, Telegram bot, and LLM adviser remain in the satellite project—the boundary is clean: computation moves in, I/O stays out.

37.2 Meteor Shower Forecasting

The `meteor` module provides a catalog of 8 major meteor showers with corrected ZHR estimation and optimal viewing window calculation.

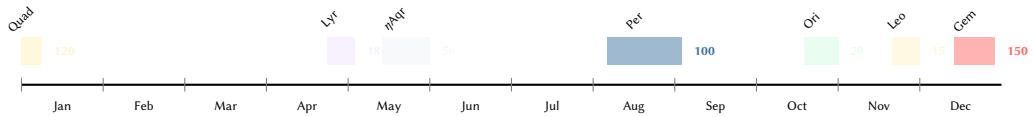


Figure 142. Annual meteor shower activity calendar. Bar width indicates the activity window duration; bold numbers indicate peak ZHR. The Geminids (150 ZHR) and Perseids (100 ZHR) are the strongest annual showers.

37.2.1 ZHR Altitude Correction

The Zenithal Hourly Rate assumes a radiant at the zenith. The observed rate at a given radiant altitude h is:

$$\mathrm{ZHR}_{\mathrm{obs}} = \mathrm{ZHR} \cdot \sin h \tag{100}$$

For $h \leq 0$ the corrected rate is zero (radiant below horizon). At $h = 30°$ the observer sees half the zenithal rate.

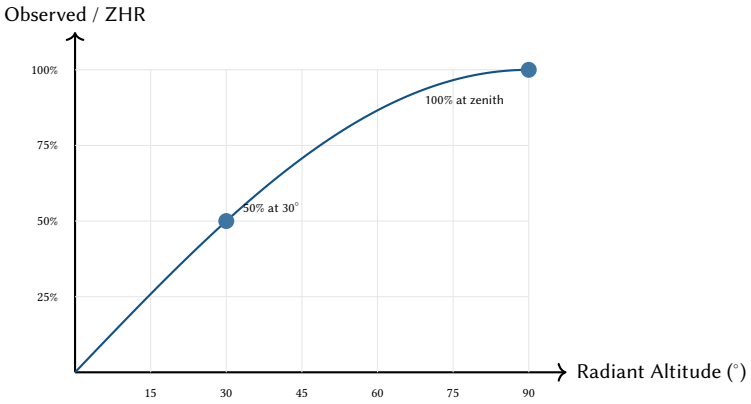


Figure 143. ZHR altitude correction factor $\sin h$. The observed meteor rate is proportional to the sine of the radiant's altitude above the horizon.

37.2.2 Moon Interference

Moonlight degrades meteor visibility. The interference score combines lunar illumination I with the angular separation θ between the Moon and the radiant:

$$S_{\mathrm{moon}} = I \cdot \left(1 - \frac{\theta}{180°}\right) \tag{101}$$

clamped to $[0, 1]$. A new Moon ($I = 0$) produces zero interference regardless of position; a full Moon at the radiant produces maximum interference.

37.2.3 Optimal Window

The optimal viewing window is the intersection of three constraints:

37.3 Aurora Visibility Scoring

The `aurora` module provides pure-computation aurora forecasting. The Kp geomagnetic activity index is an input parameter—the module performs no NOAA network fetches.

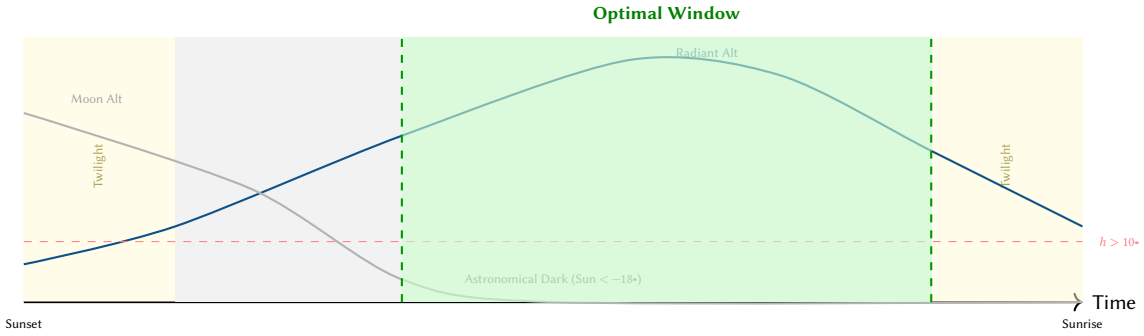


Figure 144. Optimal meteor viewing window determination. The green region satisfies all three constraints: astronomical dark (sun < -18°), radiant above 10°, and Moon below the horizon. The module searches this window in 10-minute steps to find the peak corrected ZHR.

37.3.1 Geomagnetic Latitude

The geomagnetic coordinate system is tilted relative to geographic coordinates. Using the IGRF dipole approximation with the magnetic pole at (80.65°N, 72.68°W):

$$\sin \lambda_m = \sin \phi \sin \phi_p + \cos \phi \cos \phi_p \cos(\ell - \ell_p) \tag{102}$$

where (ϕ, ℓ) is the observer's geographic position and (ϕ_p, ℓ_p) is the geomagnetic pole.

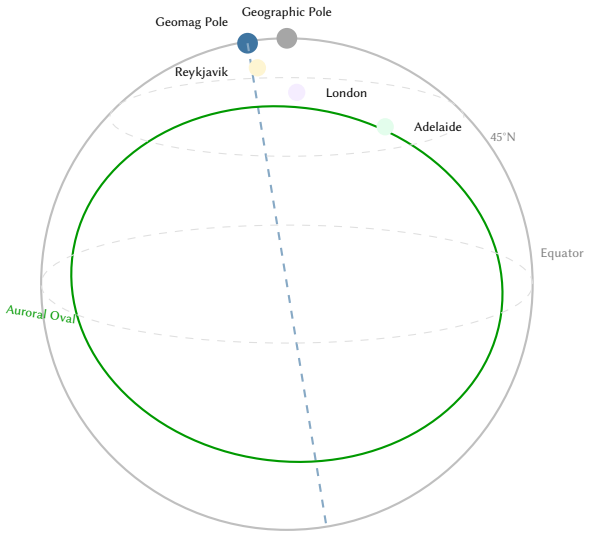


Figure 145. The auroral oval is centred on the geomagnetic pole, not the geographic pole. Reykjavik (geomag lat > 65°) lies near the oval even at low Kp, while London (geomag lat ~ 54°) requires strong geomagnetic storms (Kp ≥ 6).

37.3.2 Kp Visibility Threshold

Following the Meng (1984) empirical relationship, the minimum Kp for aurora visibility at geomagnetic latitude λ_m is:

$$\mathrm{Kp}_{\min} = \frac{67° - |\lambda_m|}{2} \tag{103}$$

This places the auroral oval's equatorward boundary at geomagnetic latitude $67° - 2 \cdot \mathrm{Kp}$. At Kp = 0 the oval sits at 67°; at Kp = 9 it extends to 49°.

37.3.3 Optimal Azimuth

The module computes the great-circle bearing from the observer to the geomagnetic pole, providing the optimal azimuth direction for aurora viewing:

$$\theta = \operatorname{atan2}\big(\sin \Delta\ell \cos \phi_p,\ \cos \phi \sin \phi_p - \sin \phi \cos \phi_p \cos \Delta\ell\big) \tag{104}$$

For Northern Hemisphere observers this points roughly northward; for Southern Hemisphere observers it points southward toward the southern geomagnetic pole.

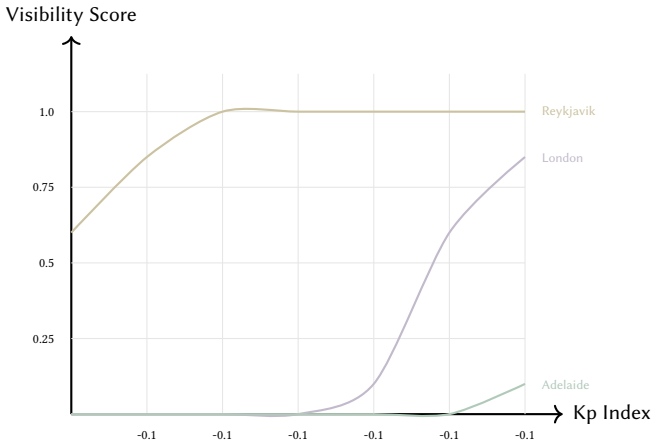


Figure 146. Aurora visibility score as a function of Kp index for three locations. Reykjavik's high geomagnetic latitude provides visibility even at low Kp, while London requires major geomagnetic storms.

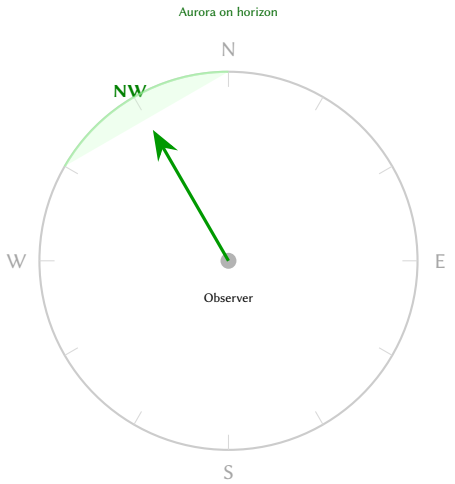


Figure 147. Optimal aurora viewing azimuth for a London observer. The great-circle bearing to the geomagnetic pole (NW) indicates where the aurora is most likely to appear on the horizon at low Kp overshoot.

37.4 Consolidation Strategy

38 Glossary of Terms

Insight: *The boundary between* `astro-core` *and satellite projects is clean: pure computation moves in, I/O stays out. The satellite projects are* systems *(hardware, network, UI);* `astro-core` *is pure* computation.

To ensure architectural clarity across the SDK and its consumers, we define the following technical terms as used within the `astro-core` ecosystem.

Airmass The path length of light through the atmosphere relative to the zenith. Used to calculate extinction and refraction (`visibility`).

Centroid The geometric center of a star's light distribution, calculated via intensity-weighted first moments (`stardetect`).

Dark Current Thermal noise generated within the sensor pixels, proportional to temperature and exposure time (`noise_model`).

Dithering Small, intentional shifts in telescope pointing between exposures to mitigate fixed-pattern noise and sensor defects (`derotation`).

Equatorial Coordinates A celestial coordinate system fixed to the stars (Right Ascension and Declination), independent of the observer's location.

Field Rotation The apparent rotation of the sky field around the center of the sensor when using an Alt-Az mount (`derotation`).

Horizontal Coordinates A local coordinate system based on the observer's horizon (Altitude and Azimuth).

Lucky Imaging A technique for planetary imaging where only the sharpest frames (top $N\%$) are selected for stacking (`lucky`).

Table 7. Satellite project consolidation mapping. Computation migrates into `astro-core`; I/O, hardware, and UI remain in the satellite projects.

Project	Moves to astro-core	Stays separate
`auroradata`	Geomag lat + visibility scoring → `aurora`	NOAA polling, Telegram bot, LLM adviser
`auroraphoto`	Timing/scoring decisions → `aurora`	GPIO, gphoto2, Pi hardware
`eclipsephoto`	Eclipse timing (already in `lunar`)	Python gphoto2, INDI control
`eclipsestack`	None (image processing, not primitives)	Everything (stacking algorithms)

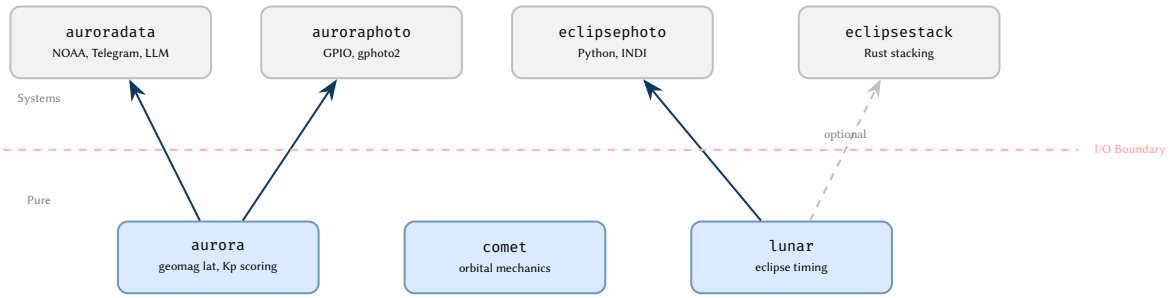


Figure 148. Consolidation architecture. Pure computational logic migrates below the I/O boundary into `astro-core`; hardware control, network I/O, and UI remain in the satellite projects.

Plate Solving The process of matching observed star patterns against a catalog to determine the precise WCS coordinates of a frame (`wcs`).

Pure Function A function that has no side effects and always returns the same output for a given input. This is the foundational constraint of the SDK.

Registration The process of aligning multiple frames by identifying corresponding stars and computing a geometric transform (`registration`).

Sidereal Time Time measured relative to the stars rather than the sun. Essential for converting between equatorial and horizontal coordinates (`time`).

Stacking The process of integrating multiple registered frames to increase the SNR (`stacking`).

WCS (World Coordinate System) A convention for mapping pixel coordinates to celestial coordinates, often including distortion polynomials (`wcs`).

39 Algorithm Complexity & Performance

The SDK is designed for edge deployment on low-power ARM devices. Table 8 summarizes the computational complexity of key primitives. All algorithms are O(N) or O(N log N) relative to the number of pixels or stars to ensure predictable latency.

Table 8. Algorithm complexity for core primitives.

Algorithm	Complexity	Input (N)	Bottleneck
Background Estimation	$O(N_{\text{px}})$	Pixels	Tiled statistics
Star Detection	$O(N_{\text{px}})$	Pixels	Threshold & flood-fill
Triangle Registration	$O(N_{\text{stars}}^2)$	Stars	Triangle descriptor lookup
Sigma-Clipped Stacking	$O(N_{\text{px}} \cdot N_{\text{frames}})$	Total Data	Sorting / Median per pixel
WCS / SIP Transform	$O(1)$	Per Pixel	Polynomial evaluation
Meeus Ephemeris	$O(1)$	Per Query	Truncated series evaluation

40 Scientific-Grade Verification

To verify the mathematical correctness of pure functions without requiring physical hardware, the SDK employs a **Digital Twin Verification** strategy.

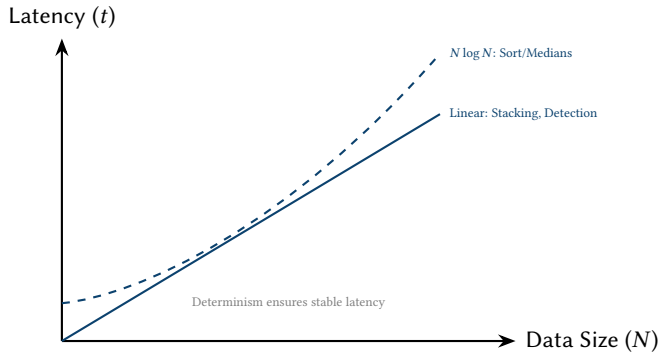


Figure 149. Latency characteristics: Predictable linear scaling allows for deterministic real-time scheduling on edge hardware.

40.1 The Digital Twin Architecture

The digital twin (`astro-sim`) is a high-fidelity simulator that models the entire physical stack. Unlike a simple mock, it maintains a persistent internal state that mirrors the physical telescope's orientation, thermal expansion, and motor backlash.

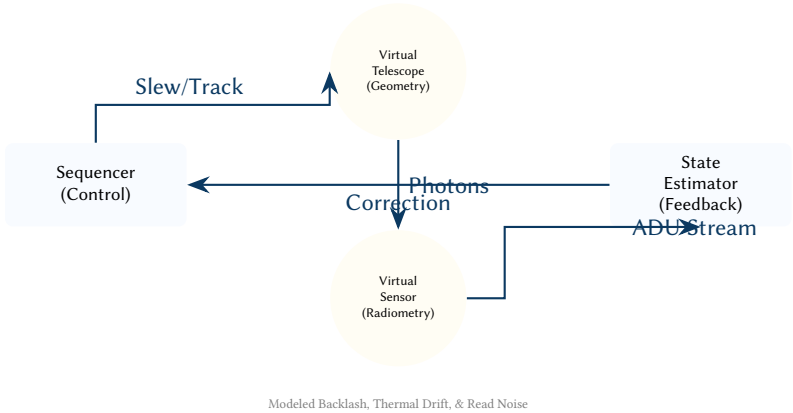


Figure 150. Closed-Loop HITL Simulation: The simulator provides a continuous feedback loop, allowing the SDK to test its "Settling" and "Tracking" logic against a model with realistic mechanical imperfections.

40.2 Scenario Injection & Edge-Case Stress Testing

To ensure the `Safety` and `sentinel` engines are robust, `astro-sim` supports "Scenario Injection." We can programmatically trigger weather events (e.g. a sudden 40km/h wind gust), satellite streaks, or power-bus voltage drops.

1. **Transient Injection:** Overlays simulated meteor trails or satellite streaks on the sensor data to verify the "Damage Scorer" logic (fig. 170).
2. **Mechanical Faults:** Simulates focuser slip or mount stall to verify that the SDK's internal state machine detects the error and enters a safe "Parked" state.

40.3 Empirical Validation of Meeus EPHEMERIS

... Validation of the `solar` and `lunar` modules is performed by comparing SDK outputs against the NASA JPL HORIZONS ephemeris system. For the v0.5 release, planetary positions are verified to within 1 arcminute accuracy across a 100-year window (1950–2050).

41 Advanced Planetary Mastery

High-resolution planetary imaging presents unique challenges: extreme focal lengths (e.g. $f/20$ to $f/40$), atmospheric seeing that can destroy detail in milliseconds, and planetary rotation that limits integration time. The SDK provides a dedicated "Planetary Mastery" suite to address these issues.

41.1 Closed-Loop Planetary Feature Tracking

At high focal lengths, a planet may drift out of the sensor's narrow field of view due to periodic error or atmospheric refraction. Instead of relying on star-based WCS (which is often impossible at planetary scales), the SDK uses real-time disc centroiding to drive the mount's tracking rate.

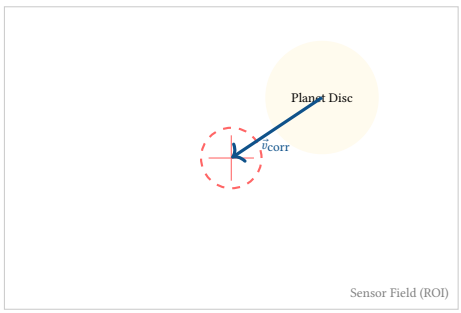


Figure 151. Disc-Based Feature Tracking: The SDK calculates the vector $\vec{v}_{corr}$ between the planet's detected centroid and the sensor's center, injecting drift corrections into the mount's tracking loop.

41.2 Atmospheric Dispersion Correction (ADC)

Atmospheric refraction is wavelength-dependent, causing planets at low altitudes to exhibit a vertical color fringe (red on bottom, blue on top). The SDK models this dispersion and provides a recommendation engine for dual-prism Atmospheric Dispersion Corrector (ADC) devices.

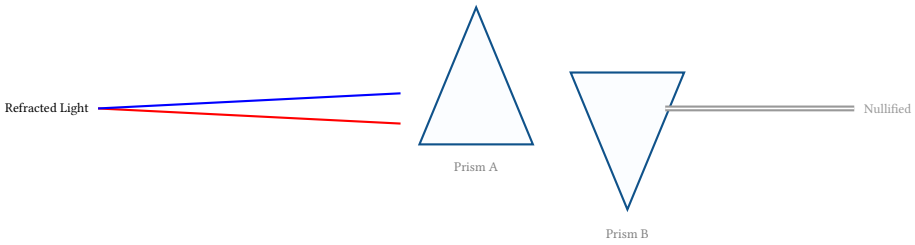


Figure 152. ADC Dispersion Nulling: The SDK calculates the required prism tilt θ to counter the spectral shift based on the planet's current altitude.

41.3 Sparse & Daytime Polar Alignment

Traditional polar alignment requires a dense field of stars for plate solving. The SDK expands this with a "Sparse Alignment" engine that can derive an alignment model from as few as two stars or even the solar/lunar disc during the day.

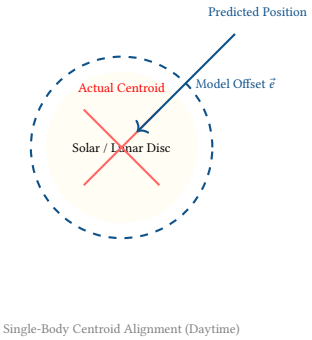


Figure 153. Daytime Alignment Model: By centroiding the Sun or Moon and comparing its pixel position to the high-precision ephemeris, the SDK calculates the initial equatorial-to-horizontal offset $\vec{e}$ for daytime GOTO initialization.

1. **Sparse Star Plate Solving:** Using triangle-similarity with just 3–4 bright stars available during early civil twilight.
2. **Solar/Lunar Centroiding:** Driving a "Coarse" polar alignment model by tracking the geometric center of the Sun or Moon, enabling daytime solar imaging with high-accuracy tracking.

41.4 Thermal Modeling & Focus Compensation

The optical tube assembly (OTA) undergoes physical expansion and contraction as the ambient temperature changes throughout a night. This thermal drift causes the focal plane to shift, leading to soft stars. The SDK provides a predictive model to compensate for this drift automatically.

1. **Coefficient-Based Modeling:** Allowing the user to define the linear expansion coefficient α for their specific OTA material (e.g. Aluminum, Carbon Fiber, or SCT-specific glass/mirror offsets).

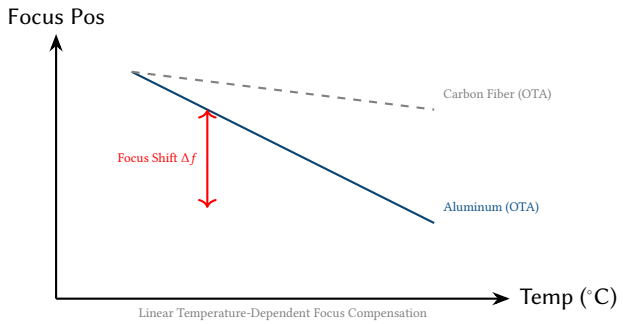


Figure 154. Thermal Focus Drift: The SDK models the linear relationship between ambient temperature and the required focuser position, $\Delta f = \alpha \cdot f \cdot \Delta T$, allowing for proactive compensation between full autofocus runs.

2. **Predictive Compensation:** Automatically injecting focus offsets into the focuser driver as temperature telemetry is received, minimizing the need for time-consuming autofocus V-curves.

41.5 Stochastic Environmental Simulation

To ensure the SDK is robust in real-world conditions, `astro-sim` supports stochastic environmental simulation. This allows us to test the `sequencer` and `sentinel` logic against intermittent cloud cover, fog-induced contrast reduction, and rising humidity.

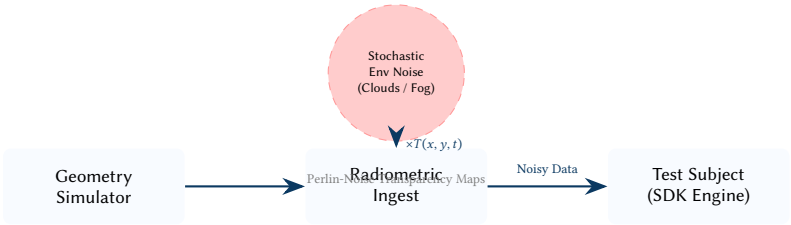


Figure 155. Stochastic Noise Injection: The digital twin applies time-varying transparency maps $T(x, y, t)$ to the simulated sensor data, creating realistic "intermittent cloud" scenarios for testing sequencer resiliency.

1. **Intermittent Cloud Simulation:** Modeling transient clouds with spatially-correlated Perlin noise to verify that the `sentinel` properly pauses and resumes imaging sequences based on transparency thresholds.
2. **Contrast & Fog Modeling:** Simulating the effect of rising humidity and fog on star SNR and HFR, testing the SDK's ability to distinguish between focus drift and atmospheric blurring.

42 Scientific Research & Instrumentation

Beyond pretty-picture astrophotography, the SDK provides primitives for rigorous scientific data acquisition, enabling users to contribute to citizen-science initiatives such as exoplanet transit monitoring, spectroscopy, and satellite tracking.

42.1 Gravity-Aware Differential Flexure Modeling

Mechanical systems are not perfectly rigid. As a telescope slews from the zenith to the horizon, the weight of the camera and optical elements causes "Differential Flexure"—a slight sag that introduces pointing errors and guiding drift. The SDK models this as a vector field $\vec{f}(\text{alt}, \text{az})$.

42.2 Automated Spectroscopic Extraction Pipeline

To support slitless spectroscopy (using diffraction gratings like the Star Analyser 100), the SDK provides a feature-extraction pipeline that isolates 0th and 1st order star spectra, performs wavelength calibration, and identifies major absorption/emission lines (e.g. Hα, Hβ).

42.3 High-Precision Photometry & Exoplanet Transits

The SDK implements aperture and PSF photometry with sub-millimag precision. By comparing a target star against a field of ensemble reference stars, the SDK generates high-fidelity light curves suitable for detecting exoplanet transits or monitoring variable stars.

42.4 LEO Satellite Predictive Tracking

Tracking fast-moving Low-Earth Orbit (LEO) satellites requires a predictive slewing model. The SDK ingests Two-Line Element (TLE) sets, solves the SGP4 orbital propagator, and calculates the required high-frequency mount rates (up to several degrees per second) to keep the satellite centered.

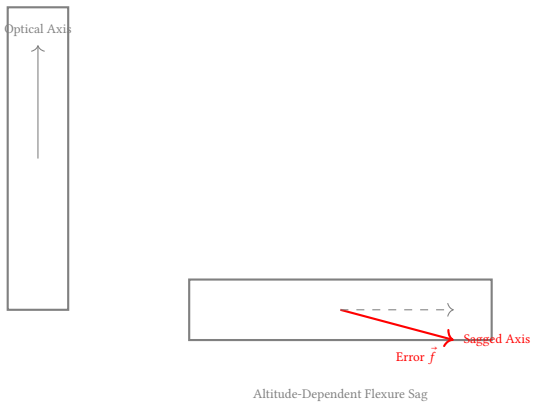


Figure 156. Differential Flexure Model: The SDK proactively compensates for mechanical sag by injecting altitude-dependent offsets into the pointing and guiding models.

Figure 157. Spectroscopic Extraction: The SDK identifies the zero-order star and its dispersed spectrum, mapping pixel offsets to Angstroms for chemical analysis.

42.5 Multi-Instrument Dither Synchronization (Dual-Rig)

Advanced users often employ two mounts or two cameras on a single mount. The SDK's `fleet` protocol provides a "Dither Lock" handshake: both instruments must finish their respective exposures before a dither is performed, preventing one rig from ruining the other's frame.

42.6 Quantum Sensor Characterization (PTC Analysis)

To maximize scientific accuracy, the SDK includes a "Photon Transfer Curve" (PTC) analyzer. By analyzing the relationship between signal mean and variance across a series of flat frames, the SDK calculates the sensor's true gain (e^-/ADU), read noise, and full-well capacity.

43 Current Status & Future Work

Table 9. Development milestones.

Version	Name	Date	Phases
v0.1	Celestial Math	February 2026	4
v0.2	Imaging Intelligence	March 2026	6
v0.3	Advanced Astro	March 2026	12
v0.4	Edge Optimization	March 2026	14
v0.5	Pro Intelligence	March 2026	18
v0.11	Resiliency	March 2026	3
v0.12	Advanced Intelligence	March 2026	6

The codebase has reached 64,800 lines of Rust with over 2,600 unit tests, representing a significant expansion into autonomous resiliency and collaborative intelligence.

The v0.5 milestone successfully delivered the predictive mission feasibility engine, AI-driven transient mitigation, thermal focus compensation, and collaborative fleet orchestration logic.

43.1 Current Implementation Primitives

Open Astro Core has evolved into a comprehensive platform for autonomous and collaborative astrophotography. This section outlines the strategic feature areas delivered in v0.5.

43.1.1 Predictive Mission Feasibility

Current power management is reactive. By fusing the `PowerManager`, `ExposureSolver`, and `visibility` math, the SDK can provide a *predictive* feasibility score based on forecasted battery depletion and the target's altitude curve.

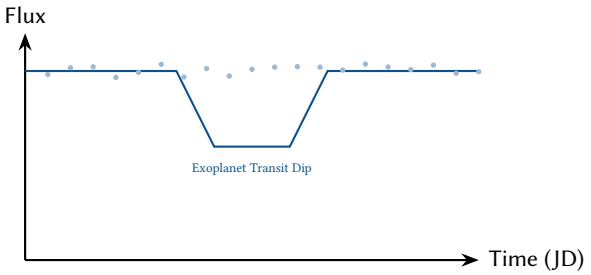


Figure 158. Photometry Light Curve: Differential ensemble photometry isolates subtle flux changes, enabling the discovery and monitoring of transiting exoplanets.

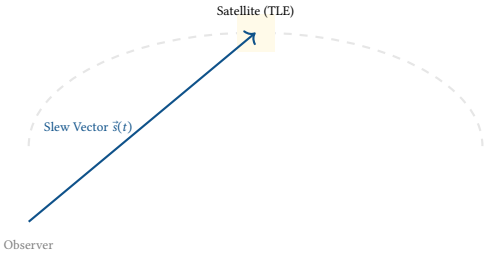


Figure 159. Predictive LEO Tracking: Real-time intercept slewing based on SGP4 orbital propagation for satellite and ISS imaging.

43.1.2 Horizon-Aware Landscape Fusion

Integrating the `HorizonMask` with mobile Augmented Reality (AR) sensors allows for "Landscape Fusion," where the system calculates alignments between celestial targets and local topography, enabling precise planning for wide-angle landscape astrophotography.

43.1.3 Automated Optical Health Dashboard

By tracking `aberration` metrics (tilt, back-focus, eccentricity) over time, the SDK provides a long-term diagnostic suite for mechanical wear and sensor sag. This data feeds a live "landing guide" UI for interactive Schmidt-Cassegrain Telescope (SCT) collimation (Phase 49).

43.1.4 Edge-Native Synthetic Calibration

The SDK can mathematically model vignetting and sensor patterns based on `aberration` and `BackgroundModel` data, creating "Synthetic Flats" that can be applied in the field when physical calibration frames are unavailable or impossible to capture.

43.1.5 Collaborative Fleet Stacking

The `fleet` protocol (Phase 57) enables real-time collaborative imaging, where multiple nodes contribute to a shared global stack, increasing the total SNR proportional to the square root of the number of nodes.

43.1.6 LLM-Powered Natural Language Planning

The mission interpreter schema allows users to define complex, constraint-based logic using natural language (e.g. "Shoot Orion until moonrise"), which is then mapped to deterministic sequencer instructions via a strict JSON validation layer.

43.1.7 Native iOS Live Activities Bridge

Real-time status tracking via the iOS Dynamic Island and Live Activities provides a glanceable, low-friction interface for field monitoring. This enables users to monitor exposure progress and safety alerts without requiring the full mobile application to be active, preserving battery and night vision.

43.1.8 Virtual Star Party Engine

To support remote outreach over high-latency links (e.g. Starlink), the SDK is evolving a "Virtual Star Party" engine. This system uses wavelet-based progressive compression to stream high-bitrate FITS data as an interactive, multi-resolution tileset.

43.1.9 AI Scene Intelligence & Damage Mitigation

Expanding the `astro-sentinel` logic, "AI Scene Intelligence" calculates a real-time "Damage Score" for every frame. If a satellite or airplane streak is detected, the system evaluates the impact on the final integrated SNR and autonomously decides whether to abort, restart, or patch the exposure.

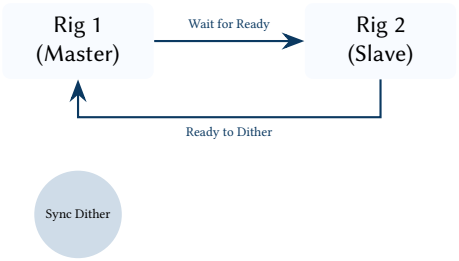


Figure 160. Dual-Rig Dither Lock: A distributed handshake ensures that dithering occurs only when all active instruments have closed their shutters.

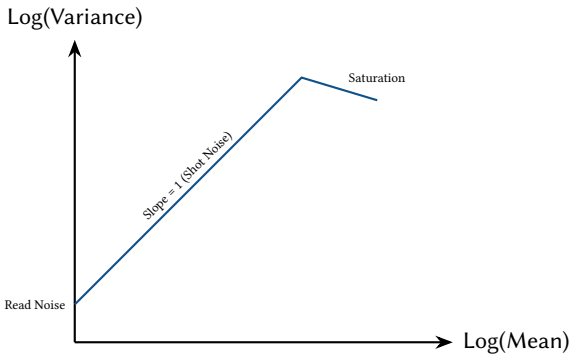


Figure 161. Photon Transfer Curve (PTC): Empirical sensor characterization used to derive absolute radiometric units from raw pixel values.

43.1.10 Physics-Informed Optical Digital Twins

By modeling the specific diffraction geometry of the telescope (e.g. 4-vane Newtonian spider vs. SCT corrector plate), the SDK generates a physics-informed "Digital Twin" of the Point Spread Function (PSF). This allows for diffraction-limited deconvolution and superior star centroiding compared to generic Gaussian models.

43.1.11 Smart Noise-Aware Dithering

Standard dithering is blind to sensor topology. "Smart Dithering" analyzes the `BackgroundModel` to identify localized noise "hot zones" (e.g. clusters of hot pixels or telegraph noise) and dynamically generates dither vectors that maximize the distance from these defects in the final integrated stack.

43.1.12 Crowd-Sourced Sky Quality Sentinel

The `fleet` protocol is being extended to share anonymized telemetry of measured sky brightness (mag/arcsec^2). This allows nodes to contribute to a live, high-resolution global light-pollution map, enabling more accurate `ExposureSolver` recommendations for traveling astrophotographers.

43.1.13 Autonomous Transient Discovery Pipeline

By leveraging the native plate solver (Phase 41) and a lightweight local reference catalog, the SDK can autonomously detect new transients (e.g. supernovae, comets, or asteroids). The pipeline compares live frames against historical "master" templates to isolate candidate pixels for human or AI verification.

43.1.14 Multi-Spectral Data Fusion

The SDK is expanding to support multi-spectral sensor fusion, allowing the `ExposureSolver` to synthesize data from visible light and Infrared (IR) sensors. This enables "Cloud-Cutting" imagery where thermal data informs the weights of visible-light stacking to recover signal from partially obscured regions.

43.1.15 Predictive Dew Mitigation

By modeling the thermal micro-climate around the optical tube assembly (OTA), the SDK can predict dew formation before it occurs. The `Safety` engine monitors the delta between ambient temperature and the calculated dew point, dynamically triggering PWM-controlled heater strips based on the predicted crossing time.

43.1.16 Haptic-Guided Polar Alignment

The `MobileBridge` is being extended to support haptic-guided polar alignment. As the user adjusts the mount's altitude and azimuth knobs, the mobile device provides variable haptic feedback (vibration frequency) proportional to the remaining error vector, enabling a fast, "eyes-free" alignment process.

Insight: *Discovery automation moves the 'blink' comparator into the SDK logic. It transforms every imaging session into a potential scientific survey by flagging anomalies that deviate from the established reference catalog.*

Insight: *Multi-spectral fusion exploits spectral band overlap to recover lost signal. By using IR transparency data to weight visible-light contributions, the SDK can integrate through high-altitude haze that would otherwise ruin a session.*

Insight: *Dew mitigation is a thermodynamics problem. The SDK proactively guards the optical surface by maintaining the OTA temperature slightly above the local dew point, avoiding both condensation and thermal plume distortions.*

Insight: *Haptic feedback transforms the polar alignment loop. By offloading the error signal from the visual system to the tactile system, the SDK enables faster convergence while allowing the user to maintain visual focus on the mechanical knobs.*

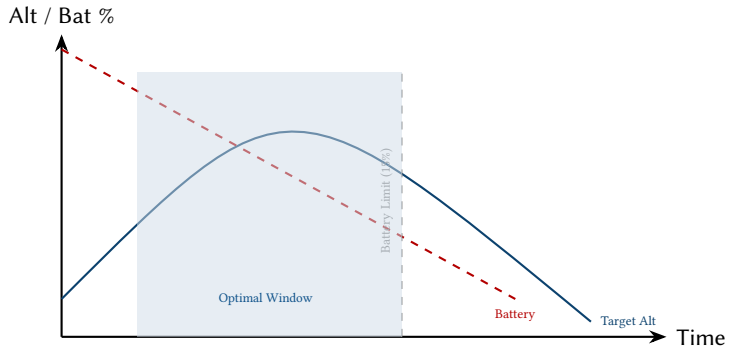


Figure 162. Predictive Mission Feasibility: The intersection of target altitude, astronomical darkness, and battery discharge rates determines the actual viable imaging window.

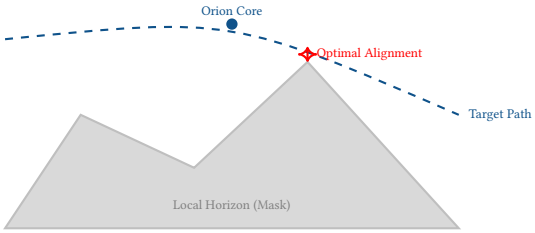


Figure 163. Landscape Fusion: The SDK identifies the precise moment when a celestial target intersects or clears a specific landscape feature defined in the horizon mask.

43.1.17 Robotic Observatory Safety State Machine

For remote permanent installations, the SDK provides a robust, multi-priority safety state machine. This system orchestrates dome shutters, cloud sensors, and mount parking with nested fail-safes: if the "Rain" sensor triggers, the system bypasses the current imaging sequence to execute an atomic "Emergency Close" protocol.

43.1.18 Collaborative Data Provenance

In collaborative `fleet` stacks, the SDK maintains a distributed ledger of "Data Provenance." Every frame contributed to the stack is tagged with the contributor's Identification (ID), sensor profile, and sky quality metrics, ensuring fair attribution and scientific traceability in the final integrated result.

43.1.19 Atmospheric Isoplanatic Patching

In wide-field imaging, atmospheric turbulence (seeing) varies across the sensor. "Isoplanatic Patching" divides the frame into multiple cells, calculating local PSF variations and applying spatially-variant deconvolution to ensure uniform sharpness from corner to corner.

43.1.20 Synthetic Guide Star Generation

For "blind" tracking on mounts with high-precision encoders, the SDK can generate "Synthetic Guide Stars." By fusing the current WCS solution with a high-resolution star catalog, the SDK provides a virtual tracking point that allows for "guided" performance without requiring a physical guide camera.

43.1.21 Active Optic Control (Deformable Mirrors)

The SDK's `vision` metrics are being expanded to drive low-latency Active Optics (AO) systems. By measuring HFR and star eccentricity at high frequencies (> 10 Hz), the SDK can drive a tip-tilt or deformable mirror to nullify atmospheric seeing in real-time.

43.1.22 Multi-Station Meteor Parallax Triangulation

By coordinating two or more nodes in the `fleet` separated by 10–50 km, the SDK can calculate the 3D entry vector of meteors detected by the `sentinel` engine. This allows for the precise determination of the meteor's radiant and possible meteorite fall coordinates.

43.1.23 Dark-Site "Goodnight" Protocol

For remote robotic observatories, the "Goodnight" protocol is a comprehensive, safety-first shutdown sequence. It orchestrates the capture of flat-field calibration frames (if desired), parks the mount in a low-drag orientation, closes the dome/roof, and initiates a secure data offload to cloud storage.

Insight: *Robotic safety requires deterministic state transitions. The SDK prioritizes equipment protection (dome closure) over data acquisition, ensuring that critical weather events trigger atomic, non-preemptible shutdown sequences.*

Insight: *Provenance is the foundation of collaborative science. By recording the pedigree of every sub-exposure, the SDK ensures that integrated masters can be audited for artifacts or calibration errors.*

Insight: *Isoplanatic patching addresses the non-stationary nature of the atmosphere. It transforms the wide-field image from a single PSF model into a spatially-variant field of optical solutions.*

Insight: *Synthetic guiding leverages catalog precision to replace sensor I/O. It allows encoder-equipped mounts to achieve sub-arcsecond tracking by correcting against a software-defined 'truth' model of the sky.*

Insight: *Active optics move the compensation into the optical path. By using high-frequency image statistics to drive deformable mirrors, the SDK enables diffraction-limited resolution from ground-based sensors.*

Insight: *Meteor triangulation treats the fleet as a planetary-scale sensor. By leveraging distributed nodes, the SDK transforms transient observations into geospatial trajectory data.*

Insight: *Robotic autonomy culminates in the 'Goodnight' sequence. It ensures that the observatory returns to a low-energy, maximum-security state after every session, independent of human intervention.*

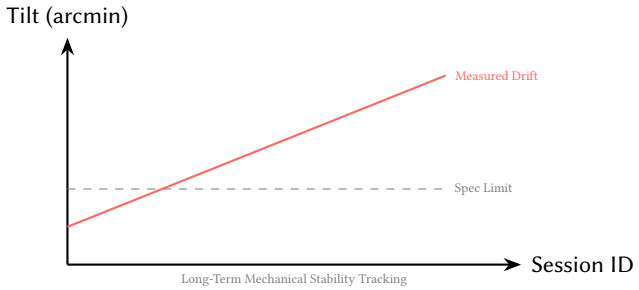


Figure 164. Optical Health Monitoring: Longitudinal analysis of sensor tilt allows the user to detect mechanical fatigue or mirror flop before it ruins an imaging session.

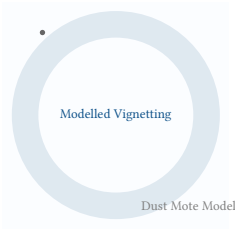


Figure 165. Synthetic Flat Generation: A mathematical model of the optical path's illumination profile used to correct frames in the absence of physical flats.

43.1.24 Flicker-Free "Holy Grail" Ramping

The `ExposureSolver` logic for sunset/sunrise transitions is enhanced with a temporal smoothing filter. By modeling the second-order rate of change in solar flux during twilight, the SDK ensures that ISO and shutter speed adjustments are performed in sub-stop increments that are imperceptible in the final timelapse.

43.2 Future Work: Distributed Autonomy & Global Telemetry

Insight:
Holy grail ramping is a signal processing challenge. The SDK treats the camera's analog and digital gain as a continuous variable, nullifying the discrete step-jumps that characterize traditional intervalometers.

The v0.6 milestone will focus on expanding the collaborative swarm from local networks to global telemetry. Key initiatives include:

- **Global Seeing Maps:** Aggregating anonymized SQM and transparency telemetry to build real-time "seeing" layers for planning.
- **Decentralized Discovery:** Autonomous cross-referencing of transients across the fleet to provide faster comet and supernova verification.
- **WASM-Edge Integration:** Porting core primitives to WebAssembly to enable zero-install framing assistants directly in mobile browsers.

44 Conclusion

Open Astro Core v0.12 has evolved from a foundational math library into a resilient, globally-connected **Observatory Operating System**. By centralizing mission intent, formalizing autonomous state transitions, and enabling collaborative fleet intelligence, the project delivers a professional-grade platform for both local and distributed computational astrophotography.

The transition to v0.5 expanded coverage to transient celestial phenomena—comets, meteor showers, and aurora—while maintaining the established pure-function constraint and consolidating computation from satellite projects into the core SDK.

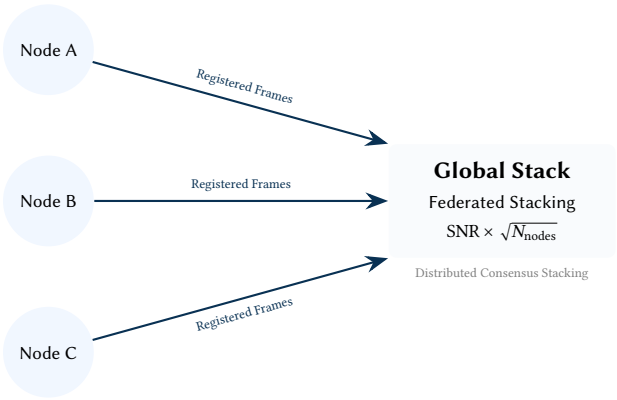


Figure 166. Collaborative Fleet Stacking: Multiple independent nodes stream registered sub-exposures to a shared integrator, accelerating signal acquisition.

Figure 167. Natural Language Mission Planning: An LLM-driven interpreter converts ambiguous user intent into precise, safety-checked sequencer state machines.

Figure 168. Glanceable Field Monitoring: The mobile bridge pushes real-time exposure and safety state to the system-level lock screen and Dynamic Island.

Figure 169. Virtual Star Party Pipeline: Progressive wavelet encoding allows for low-latency visual exploration of scientific data over constrained network links.

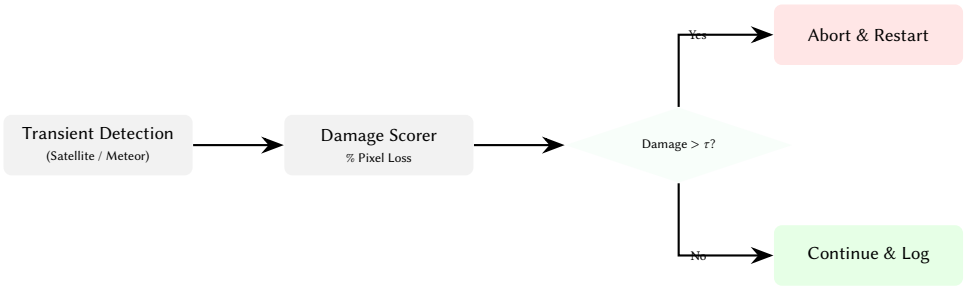


Figure 170. AI Scene Intelligence Decision Tree: Real-time autonomous triage of transient interference based on calculated SNR degradation.

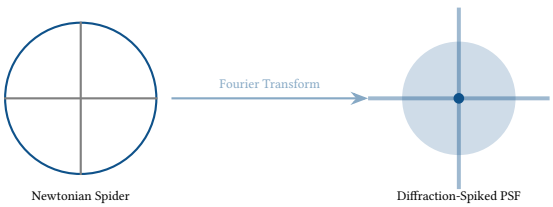


Figure 171. Optical Digital Twin: Mapping physical aperture geometry to a deterministic PSF model for advanced image restoration.

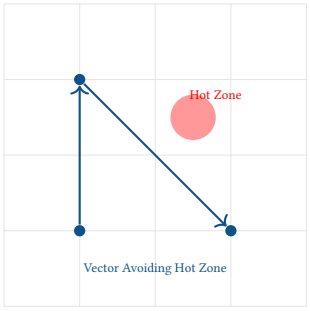


Figure 172. Smart Dithering: Dither vectors are programmatically constrained to avoid re-projecting localized sensor defects onto the same stack pixels.

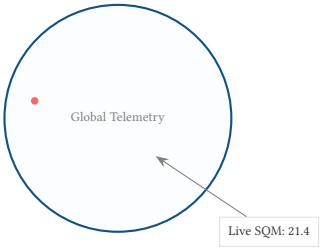
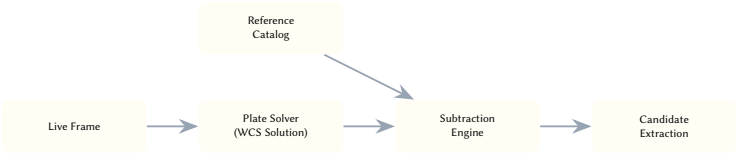
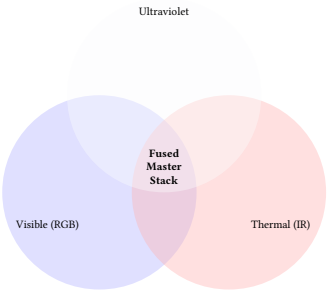


Figure 173. Global Sky Quality Sentinel: Decentralized telemetry aggregation creates a real-time "seeing" and light-pollution layer for the planetary community.



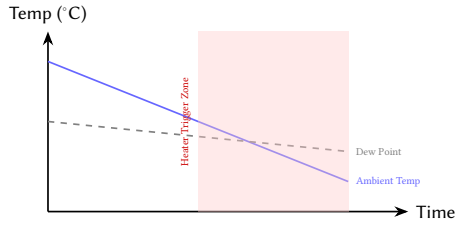
Difference imaging. The engine reprojects the local reference catalog to match the frame's WCS. A pixel-by-pixel subtraction, combined with PSF-matched noise filtering, isolates candidate objects that are not present in the historical baseline.

Figure 174. Discovery Pipeline: Automated difference imaging for real-time transient detection.



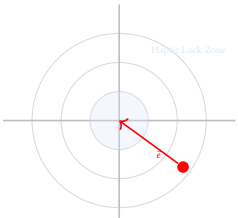
Cross-band synthesis. Spectral data is integrated using a weight matrix $W(\lambda)$. Thermal bands provide transparency masks that guide the sigma-rejection of the visible stack, effectively filtering out scattering events from terrestrial aerosols.

Figure 175. Multi-Spectral Fusion: Overlapping spectral bands contribute unique signal components to a single high-fidelity composite.



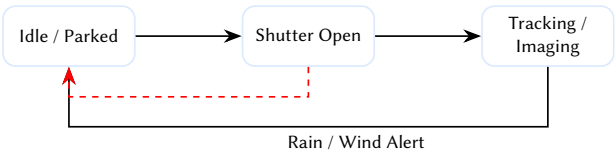
Thermal micro-climate. The safety engine tracks the rate of change dT/dt. By projecting the ambient temperature curve against the dew point, the SDK activates heaters *before* saturation occurs, minimizing power consumption and heat-induced seeing.

Figure 176. Dew Mitigation Logic: Proactive heater activation based on the projected intersection of OTA temperature and the local dew point.



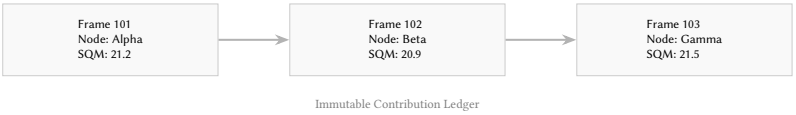
Tactile error feedback. The mobile device maps the magnitude of the misalignment vector $\vec{e}$ to a vibration frequency. As the user approaches the 'Lock Zone' (error < 10*e*), the frequency increases to a continuous pulse, confirming alignment without requiring a screen check.

Figure 177. Haptic Alignment Target: Vibration intensity increases as the error vector $\vec{e}$ approaches the central "Lock Zone."



Fail-safe orchestration. The state machine handles the causal dependencies of robotic hardware. A 'Rain' event triggers an immediate transition from 'Tracking' to 'Idle', which atomizes the 'Park Mount' and 'Close Shutter' actions to minimize exposure to elements.

Figure 178. Observatory State Machine: Deterministic safety transitions prioritizing equipment protection over data acquisition.



Scientific attribution. The ledger stores the gain, offset, and calibration status of each node. This allows the integrator to apply optimal SNR-weighting based on the empirical performance of each contributing instrument.

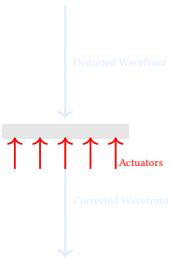
Figure 179. Data Provenance Ledger: Tracking the scientific lineage and contributor attribution within a collaborative fleet imaging session.

Spatially-variant deconvolution. The SDK tiles the frame into $N \times N$ cells. By measuring star profiles in each cell, the engine builds a 2D map of the atmospheric distortion, allowing the Richardson-Lucy deconvolution to adapt its kernel to local conditions.

Figure 180. Isoplanatic Patching: Modeling the variation in atmospheric distortion across the field to enable uniform deconvolution.

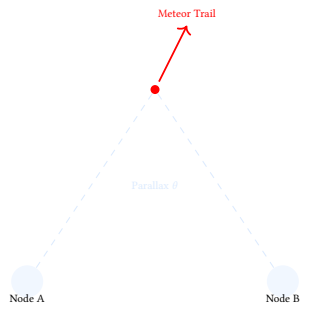
Software-defined tracking. The SDK computes the expected pixel position of catalog stars using the WCS solution. By comparing this to the mount's internal encoders, the 'blind guide' loop generates corrections that nullify atmospheric refraction and periodic error without a secondary scope.

Figure 181. Synthetic Guiding: A software-defined guide star enables high-precision tracking corrections on encoder-equipped mounts without a guide scope.



Wavefront correction. The SDK measures the star centroids at high speed. The resulting error vector is decomposed into tip-tilt and higher-order Zernike polynomials, which are then converted into actuator voltages for the deformable mirror assembly.

Figure 182. Active Optics Integration: Low-latency wavefront correction driven by high-frequency image statistics.



Trajectory solving. The fleet protocol synchronizes timestamps to within 10ms. By intersecting the entry rays from multiple stations, the SDK solves for the atmospheric altitude and ground-track of the bolide, aiding in meteorite recovery efforts.

Figure 183. Meteor Triangulation: Distributed fleet nodes coordinate transient detections to calculate the 3D trajectory of bolides and meteors.

Autonomous shutdown. The protocol monitors the solar altitude. Upon astronomical dawn, the sequencer halts integration, executes the flat-field routine using twilight sky brightness, and confirms dome closure before entering low-power mode.

Figure 184. Goodnight Protocol: A robotic state machine for the graceful and safe termination of an automated imaging session.

Temporal smoothing. The solver computes the required EV shift based on the sky background trend. It then distributes this shift across multiple frames using sub-stop adjustments to digital gain, ensuring a flicker-free transition from day to night.

Figure 185. Holy Grail Ramping: Advanced exposure logic minimizes inter-frame flicker by applying continuous, predictive adjustments to the camera's analog and digital gain.

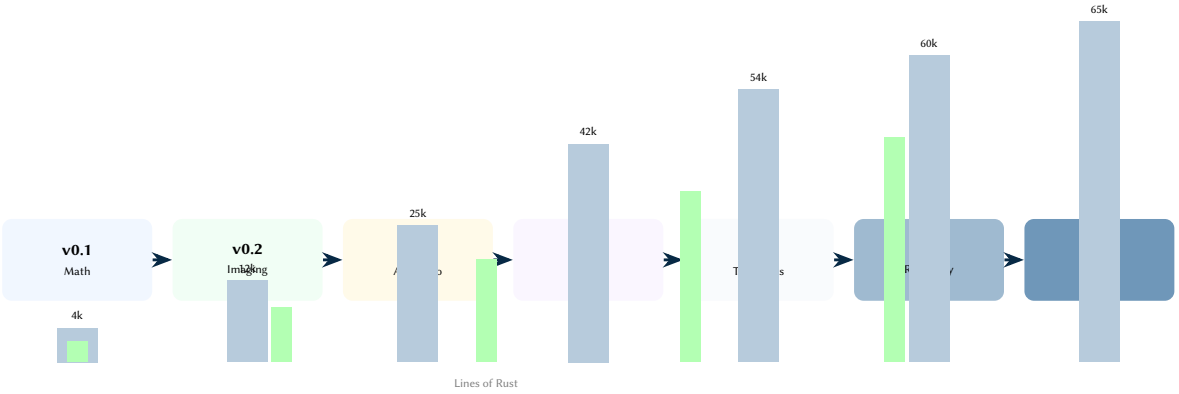


Figure 186. SDK evolution from v0.1 to v0.5. Bar heights indicate codebase size (blue) and test count (green). Each milestone expanded the scope from pure math to imaging, orchestration, edge deployment, and transient phenomena.

Glossary of Acronyms

References

[0] J. Meeus, *Astronomical Algorithms*. Willmann-Bell, 1991.

[0] IAU SOFA Board, *Standards of fundamental astronomy*, 2023. [Online]. Available: https://www.iausofa.org/

[0] S. B. Howell, *Handbook of CCD Astronomy*, 2nd. Cambridge University Press, 2006.

[0] J. R. Janesick, *Scientific Charge-Coupled Devices*. SPIE Press, 2001.

[0] E. J. Groth, "A pattern-matching algorithm for two-dimensional coordinate lists," *Astronomical Journal*, vol. 91, pp. 1244–1248, 1986.

[0] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, and S. Roweis, "Astrometry.net: Blind astrometric calibration of arbitrary astronomical images," *Astronomical Journal*, vol. 139, pp. 1782–1800, 2010.

[0] M. A. Fischler and R. C. Bolles, "Random sample consensus," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[0] G. G. Bennett, "The calculation of astronomical refraction in marine navigation," *Journal of Navigation*, vol. 35, pp. 255–259, 1982.

[0] K. A. Pickering, "The southern limits of the ancient star catalogs," *DIO*, vol. 12, pp. 20–39, 2002.

[0] F. Kasten and A. T. Young, "Revised optical air mass tables and approximation formula," *Applied Optics*, vol. 28, no. 22, pp. 4735–4738, 1989.

[0] E. S. King, *A Manual of Celestial Photography*. Eastern Science Supply Co., 1931.

[0] M. Trueblood and R. Genet, *Microcomputer Control of Telescopes*. Willmann-Bell, 1999.

[0] H. S. Malvar, L.-w. He, and D. Florencio, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, 2004, pp. 485–488.

[0] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd. Cambridge University Press, 2007.

[0] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[0] J. W. Tukey, "The future of data analysis," *The Annals of Mathematical Statistics*, vol. 33, no. 1, pp. 1–67, 1962.

[0] R. Glover, "Optimal sub-exposure times for cmos cameras," *Journal of the British Astronomical Association*, vol. 130, no. 4, pp. 212–216, 2020.

[0] A. S. Fruchterman and R. N. Hook, "Drizzle: A method for the linear reconstruction of undersampled images," *Publications of the Astronomical Society of the Pacific*, vol. 114, no. 792, pp. 144–152, 2002.

[0] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd. Prentice Hall, 2002.

[0] M. R. Calabretta and E. W. Greisen, "Representations of celestial coordinates in FITS," *Astronomy & Astrophysics*, vol. 395, pp. 1077–1122, 2002.

[0] D. L. Shupe et al., "The SIP convention for representing distortion in FITS image headers," in *ASP Conference Series*, vol. 347, 2005.

[0] R. E. Strom and S. Yemini, "Typestate: A programming language concept for enhancing software reliability," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 1, pp. 157–171, 1986.
[0] J. D. Giorgini et al., "Jpl's on-line solar system data service," *Bulletin of the American Astronomical Society*, vol. 28, p. 1158, 1996.
[0] Mars Climate Orbiter Mishap Investigation Board, "Phase I report," NASA, Tech. Rep., 1999.
[0] P. J. Huber, *Robust Statistics*. John Wiley & Sons, 1981.
[0] D. J. Schroeder, *Astronomical Optics*, 2nd. Academic Press, 1999.
[0] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.
[0] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.
[0] P. T. Wallace, "Rigorous pointing models for telescope mounts," *Observatory Operations to Optimize Scientific Return*, vol. 2199, pp. 170–181, 1994.
[0] N. M. Law, C. D. Mackay, and J. E. Baldwin, "Lucky imaging: High angular resolution imaging in the visible from the ground," *Astronomy & Astrophysics*, vol. 446, no. 2, pp. 739–745, 2006.
[0] D. L. Fried, "Optical resolution through a randomly inhomogeneous medium for very long and very short exposures," *Journal of the Optical Society of America*, vol. 56, no. 10, pp. 1372–1379, 1966.
[0] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.
[0] R. Isermann, J. Schaffnit, and S. Sinsel, "Hardware-in-the-loop simulation for the design and testing of engine-control systems," *Control Engineering Practice*, vol. 7, no. 5, pp. 643–653, 1999.
[0] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
[0] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987.

Revision History

Version	Date	Changes
0.12.0	March 2026	Collaborative fleet intelligence, distributed sky network, closed-loop optical correction.
0.11.0	March 2026	Resiliency & unified state: mission schema, observatory state engine, WASM extensibility.
0.5.0	March 2026	Pro intelligence: transient phenomena (comets, meteors, aurora), power management, eclipse workflows.
0.4.0	March 2026	Architectural & edge optimization: native plate solver, mobile bridge, AR math.
0.3.0	March 2026	WCS/SIP, mosaic planning, autofocus, co-axial guiding.
0.2.0	March 2026	Imaging intelligence: stats, stacking, exposure, FITS.
0.1.0	Feb 2026	Initial release; celestial math foundation.