Smart India Hackathon 2026

Problem Statement ID 26169 | Theme: Smart Automation, Space Technology | Category: Software

Organisation: Department of Space, Indian Space Research Organisation, Space Applications Centre

# Technical Report

## Development of an AI-Based Virtual Camera Tracking System for Coarse Alignment of Mobile Free Space Optical Communication (FSOC) Terminals

ARGUS

Acquire, Recognise, Guide, Update, Stabilise


**Team name** BlankPoint

**Team ID** 182144

**Institute** Dayananda Sagar College of Engineering (DSCE), Bengaluru

**Team leader** Shivansh Pandey

**Team members**

Govind Pandey Tanmay Tiwari Manya Kumar
Abhijeet Yadav Lakshita Didwania Shivansh Pandey



**Abstract.** The problem statement asks for a software platform that replaces a camera, a pan-tilt mount and optics for developing the coarse alignment stage of FSOC pointing, acquisition and tracking. ARGUS is that platform: a configurable virtual scene with one or more moving beacons, a virtual camera with realistic rate limits, a chain of atmospheric, platform and sensor disturbances, and an automatic detector, estimator and controller that keep the designated beacon in the camera window. Detection is a matched filter with a sub-pixel Gaussian fit; a track-before-detect path finds beacons at three to six sigma; an interacting multiple model estimator predicts motion; a five-state machine holds identity among decoys; a feed-forward plus PI controller drives the camera. A small convolutional network fills gaps in degraded conditions. Every run writes a per-frame log, a summary and a PDF report with the metrics of *PS row 16 to 20*. On the 17-scenario regression pack the tracker meets every performance limit in clear, noisy, foggy and low-light conditions; the two settings it does not meet (platform shake at the PS maximum, hard mode) are physical limits, measured and documented. The desktop application runs offline on Windows, Linux and macOS and reads the evaluators' videos for Benchmark 2.

# Contents

**1 Problem understanding** **2**
1.1 Context . . . 2
1.2 What the problem statement asks . . . 2
1.3 Reading of the specification . . . 3
1.4 Points the problem statement leaves open, and the choice made . . . 3
1.5 Numbers that shape the design . . . 4

**2 System architecture** **4**

**3 Software modules** **4**

**4 Tracking methods** **5**
4.1 Detection and centroiding . . . 5
4.2 Estimation . . . 6
4.3 State machine, identity and designation . . . 6
4.4 Control . . . 7

**5 AI methods** **7**

**6 The application** **8**

**7 Test methodology** **9**

**8 Performance analysis** **10**

**9 Limitations and future improvements** **12**

**10 Conclusion** **13**

**A Scenario pack** **14**

**B Performance log: metric definitions printed in every report** **15**

**C Problem statement parameter table, row by row** **15**

# 1 Problem understanding

## 1.1 Context

FSOC links carry data on a highly directional laser beam. Against radio they offer gigabit-to-terabit rates, licence-free operation and immunity to electromagnetic interference [2, 3], but a small angular error breaks the link. Between moving platforms the terminals run a pointing, acquisition and tracking (PAT) loop [1, 4]: a coarse stage brings the remote terminal's beacon into the camera field of view and holds it there, and a fine stage locks the beam. This work addresses the coarse stage. Developing it on hardware needs a camera, a pan-tilt mechanism and optics; the problem statement asks for software that replaces them.

## 1.2 What the problem statement asks

The functional objective is a system that "autonomously detects, identifies, and continuously tracks a designated moving target within a virtual scene by controlling a virtual camera viewport". Eight

"shall" functions follow: generate a configurable environment, generate one or more moving targets, implement a movable camera, detect the beacon automatically, track it with computer vision, control and reposition the camera, generate and introduce disturbances into the camera feed, and display performance in real time. The parameter table (rows 1 to 25) fixes the scene, camera, target, motion, performance and disturbance values; the deliverables are the application, the source, this report, the user manual and an automatically generated performance log. Evaluation has four stages: functional verification, Benchmark 1 (the evaluators' scenarios), Benchmark 2 (their 30 fps videos, with the PTZ camera bypassed) and the technical evaluation.

### 1.3 Reading of the specification

- The screen (at least $2000 \times 2000$ px, *PS row 1*) is what the tracker observes every frame: the "simulated video stream". The $640 \times 480$ px camera window ($4° \times 3°$, *PS row 3, 4*) is the controlled output and represents where the terminal points. It starts at the screen centre (*PS row 6*) and turns at 5 to 10 deg/s (*PS row 13, 14*).
- With one screen pixel equal to one camera pixel the instantaneous field of view is $4°/640 = 0.00625°$ (22.5 arcsec) per pixel. The 10 px tracking limit (*PS row 17*) is 0.0625°; at 5 deg/s and 30 Hz the window moves at most 26.7 px per frame; the maximum platform motion of 20 px per frame (*PS row 25*) therefore consumes 75 % of the slew budget.
- Benchmark 2 replaces the simulated scene with the evaluators' video. The tracker and camera control run unchanged on its frames; the output is the per-frame centroid and the timing and lock metrics. With the evaluators' positions as a truth file, tracking error, centroiding error, RMSE and true lock retention are computed against them.
- Two error terms appear in the specification and both are logged: tracking error (true beacon to window centre) and centroiding error (measured to true centroid).

### 1.4 Points the problem statement leaves open, and the choice made

| Question | Choice | Reason |
|---|---|---|
| Does the tracker see the whole screen or only the window? | Whole screen by default; a hard mode restricts it to the window | "Observe the surrounding environment", "simulated video stream"; a blind sweep at 5 deg/s cannot reach a far corner within 2 s |
| Screen pixels to degrees | One screen pixel equals one camera pixel: 12.5° across 2000 px | The only anchor given is 4° over 640 px |
| Meaning of "lock" | TRACK state with the estimate within 30 px of the window centre | Acquisition needs a capture criterion; 20 px lowered retention under vibration without lowering error |
| A sustained 20 px per frame platform shift | A bounded sway with that peak speed, amplitude at most 20 % of the screen | A sustained shift leaves the screen in seconds |
| How the target is "designated" (*PS row 8*) | By appearance, by a start cue (as ephemeris or an operator would give) or by a point; a video's beacons are listed for the operator to choose | The PS does not say; identical look-alikes cannot be told apart by appearance alone |
| Role of AI | Classical detector first; a CNN fills gaps and targets faint beacons, trained on the simulator's exact labels | Keeps frame rate and reliability independent of the model |

### 1.5 Numbers that shape the design

| Quantity | Value | Derivation |
|---|---|---|
| Angular size of one pixel | 22.5 arcsec | 4°/640 px |
| Allowed tracking error | 0.0625° | 10 px × 22.5 arcsec |
| Maximum window motion | 26.7 px per frame | 5 deg/s / 30 Hz / 0.00625° per px |
| Platform motion at the PS maximum | 20 px per frame, 75 % of the slew budget | *PS row 25* against *PS row 13* |
| Window-sized areas per screen | about 13 | (2000/640) × (2000/480) |
| Centre to corner with the target known | 0.95 s | max(4.25°, 4.75°)/5 deg/s |

## 2 System architecture

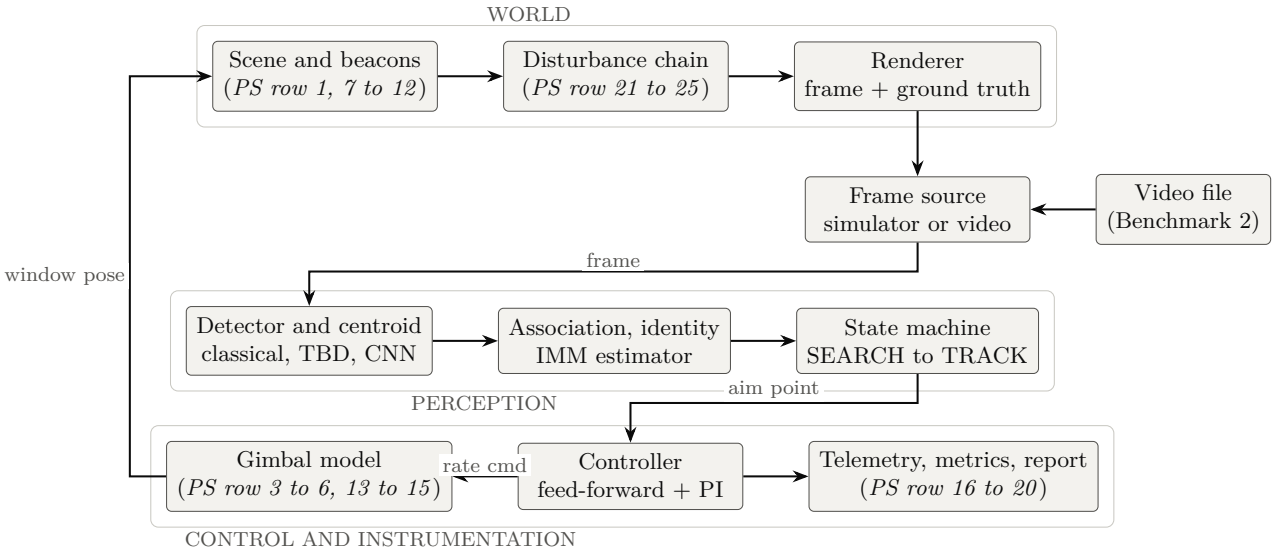



**Figure 1:** Block structure (rate cmd: the pan and tilt rate command). One `Simulation` object drives every frame for the desktop application, the web application, the command line and the tests, so all execute the same code path. The frame source yields frames from the simulator or from a video file; nothing downstream knows which.

Data flow per frame: the kinematics advance; the gimbal integrates the previous command; the renderer draws the scene and applies the disturbance chain; the tracker detects, measures, associates and updates the estimator; the state machine chooses what to aim at; the controller emits a rate command clipped by the gimbal model; a telemetry record is written. The loop is a fixed step at the camera update rate and is deterministic from the scenario file and the seed, so any run can be repeated exactly.

## 3 Software modules

| Module | Purpose |
|---|---|
| `engine/config.py` | Typed configuration, one field per PS row; YAML load and save; clamping of every numeric input. |
| `world/scene.py`, `sprites.py` | Backgrounds (starfield, terrain, gradient, flat); beacon shapes (square, circle, Gaussian, cross, ring, diamond, custom mask) at width × height. |

| Module | Purpose |
|---|---|
| `world/targets.py` | Beacon kinematics: line, circular, figure of 8, random (Ornstein-Uhlenbeck), spiral, sinusoidal, user-defined waypoints, static; optional blink code. |
| `world/camera.py` | Gimbal: pose in degrees, rate and acceleration limits, command latency, IFOV conversions. |
| `world/disturbance.py` | Extinction, turbulence (wander, scintillation), PSF blur, platform sway, vibration, exposure gain, frame loss, Poisson, Gaussian and salt-and-pepper noise, applied in physical order with independent random planes. |
| `world/renderer.py` | Draws the scene with sub-pixel beacon placement and returns the ground truth. |
| `perception/detect.py` | Classical detector (median, background subtraction, matched filter, adaptive threshold, connected components, shape filters), sub-pixel centroid (centre of gravity plus 2-D Gaussian fit), CNN heat-map detector (ONNX). |
| `perception/estimator.py` | Interacting multiple model over constant velocity, constant acceleration and coordinated turn. |
| `perception/egomotion.py` | Frame-to-frame picture shift by phase correlation. |
| `control/tracker.py` | State machine SEARCH, VERIFY, TRACK, COAST, REACQUIRE; gating; appearance signature; beacon code check; adaptive measurement noise. |
| `control/controller.py` | Feed-forward plus PI rate controller with latency lead, deadband and anti-windup. |
| `engine/simulation.py`, `metrics.py`, `checks.py`, `report.py` | Run loop and telemetry record; metric definitions with pass or fail; scenario check (clamped, beyond the PS, near a limit, physically impossible); PDF report. |
| `gui/app.py`, `webapp/`, `cli.py` | Desktop application (PyQt6), web application (same panel from `ui_shared.py`), commands `run`, `video`, `batch`, `verify`, `gui`. |

# 4 Tracking methods

## 4.1 Detection and centroiding

A $3 \times 3$ median ($5 \times 5$ under heavy salt and pepper) removes specks. A $31 \times 31$ box blur estimates the smooth background, which is subtracted in signed 16-bit arithmetic: an unsigned subtraction clips the negative half and buried faint beacons. A Gaussian matched filter at one third of the configured beacon size raises the signal-to-noise ratio of an extended spot against point noise and stars. The noise level is the median absolute deviation of the filtered image, so stars and the beacon itself cannot inflate the threshold. Pixels above level $+ k\sigma$ ($k = 4$) are labelled by connected components; blobs are kept if area, aspect ratio and fill are consistent with a spot, and scored by matched-filter SNR, peak and agreement with the designated beacon's expected size. The leading candidates are refined by an intensity-weighted centre of gravity followed by a 2-D Gaussian least-squares fit [9], giving centroids to about 0.01 px on clean frames and 0.1 to 0.2 px under heavy noise. Among several acceptable candidates the designated beacon is chosen by the fitted width expected from its configured size and shape (*PS row 9, 10*), a measure that does not move with the noise level, then by brightness.

**Faint beacons (track-before-detect).** After extinction a beacon can sit at three to six sigma per frame, where a single-frame threshold either misses it or floods the tracker with noise. When no candidate reaches the acquisition confidence, the detector runs at $k = 3$ on a moving-target residual (the frame minus a running mean of the picture, which holds the stars and the sky but not a moving beacon), and the weak candidates are linked frame to frame into chains with a consistent velocity [8]. A chain seen in six of the last eight frames with mean SNR above 3.5 and a fitted width near the designated beacon's is promoted to a provisional track that seeds the estimator with the chain's velocity and must survive a stricter verification. Two guards keep a faint track on the beacon: the raw detection is kept when the sub-pixel refit slides more than 3 px or its width exceeds 2.5 times

the recent median, and the CNN is not used on a faint track. The chain linking is vectorised with NumPy; on the faint scenario the 99th-percentile frame time fell from about 150 ms to about 30 ms.

### 4.2 Estimation

An interacting multiple model (IMM) estimator [6, 7] runs three Kalman filters (constant velocity, constant acceleration, coordinated turn) and blends them by likelihood. Process noise is sized for the accelerations the PS produces (beacon paths to about 100 px/s$^2$, platform sway to about 600 px/s$^2$). Measurement noise adapts: the zero-mean part of the innovation sequence is treated as vibration and added to the measurement variance, so the estimator smooths jitter instead of chasing it, while a consistent innovation (lag) is not mistaken for noise.

### 4.3 State machine, identity and designation

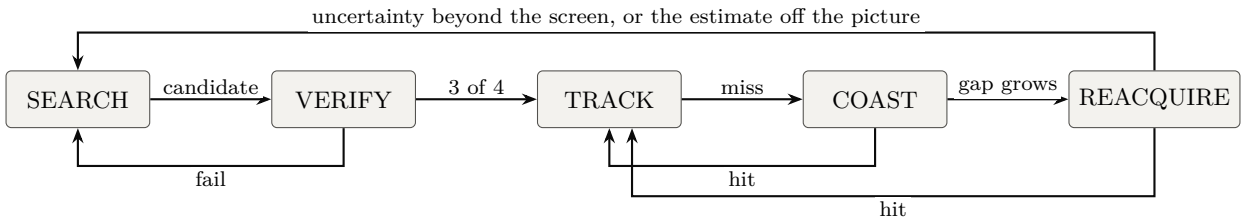



**Figure 2:** The five-state tracker. VERIFY needs three consistent frames of four ("3 of 4") before TRACK, so a noise speck is never counted as acquisition. In hard mode SEARCH drives an outward square spiral of the window.

SEARCH detects on the whole observed picture. TRACK searches a region around the prediction, gates candidates by Mahalanobis distance and rejects those whose appearance signature (area, peak, PSF width) differs strongly from the beacon being followed, which holds identity through crossings with decoys. COAST propagates the prediction through short dropouts; REACQUIRE widens the search with the growing uncertainty. A coasting guard returns to a whole-scene search when the estimate lies more than the search window outside the picture; without it a coasting estimate drifted off the screen on a faint run (centroid error to 1268 px, lock 75 %; with it 5.9 px and 96 %).

**Designation** (*PS row 8*, "a designated moving target"). Every target has a name; all are detected, the designated one is followed and scored. Mode *appearance* picks it by configured shape, size and

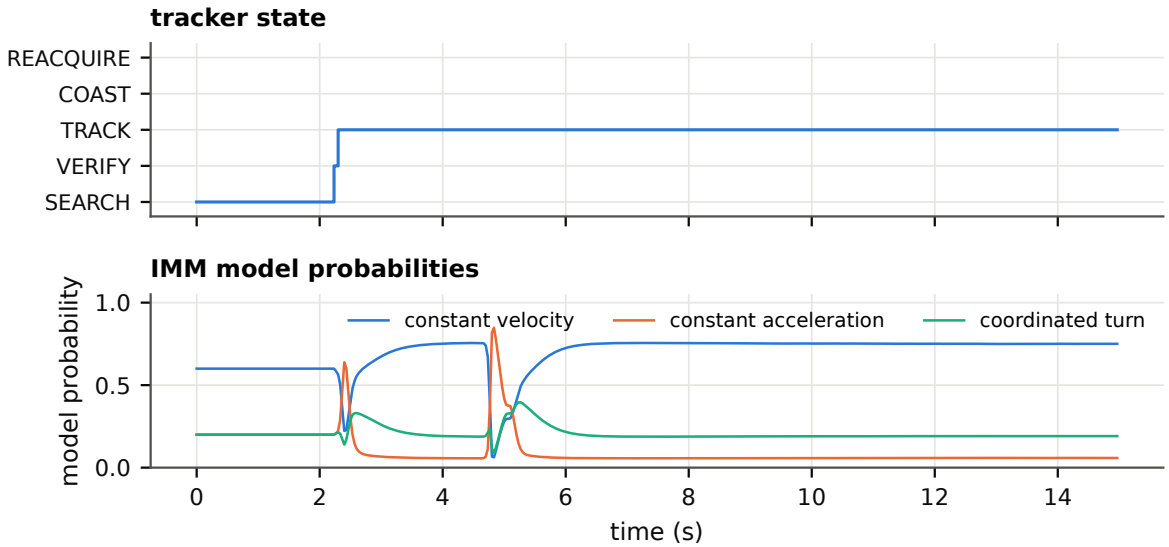



**Figure 3:** A hard-mode run (tracker restricted to the camera window): the window sweeps in SEARCH until the beacon enters it, VERIFY confirms it, and TRACK holds; below, the IMM model probabilities as the estimator settles on constant velocity for a straight path.

brightness; *start* also gives its start position, as an operator or an ephemeris cue would; *cue* gives a point near it. The automatic mode uses the start cue whenever targets start apart and appearance when they start together. On loading a video the detector scans the first 30 frames and lists the beacons found as Target 1, 2, ... in the panel; the operator chooses one and the tracker is cued to its first-frame position. In appearance mode, search frames where another spot scored within 0.15 of the chosen one are reported as ambiguous frames. Tracking every beacon at once was not built: the PS metrics are for one target and one camera.

**Coded beacon** (beyond the PS). A target may blink a bit pattern at up to half the frame rate; a 0 bit dims it to 70 %. When the designated target has a code, the tracker samples the followed spot's brightness every frame and correlates the last two code cycles with the code at every phase. VERIFY ends in TRACK only when the best correlation is at least 0.6 with the 0 bits at least 12 % dimmer; a spot showing no modulation is turned down and not picked again for 3 s. With three identical look-alikes and no cue, the coded beacon is followed 100 % of the locked time over three seeds; without the code the same runs follow a look-alike on two seeds.

**Scenario check.** Every numeric input is clamped in the engine, so no front end can pass an impossible value. `engine/checks.py` then notes values beyond the PS with the row named, near-limit settings (a beacon above 70 % of the camera turn rate) and physical limits (a beacon faster than 800 px/s at 5 deg/s; jitter plus platform above 26.7 px per frame; look-alikes in appearance mode). The notes appear in the panel, at Start, in the end dialog and on page 1 of the report.

### 4.4 Control

The angular error is the IFOV times the pixel offset from the window centre to the target position led by the command latency, with the window's own motion led likewise. The command is feed-forward of the target angular rate plus PI on the error ($k_p = 5$, $k_i = 0.8$), with a 1.5 px deadband and integrator clamping while saturated. The derivative gain is 0: at 0.3 it acted on an error that reaches the controller one frame late and in whole pixels, and drove a limit cycle of about ±10 px on a still beacon (mean 6.4 px); with $k_d = 0$ a still beacon is held within 4 px (mean 1.6 px), and tracking errors over the pack dropped by about two thirds. The velocity lead is capped so the path turns by at most 0.1 rad over it: a straight extrapolation pointed the feed-forward off a tight curve, giving 34 px error and 14 to 19 % lock on a 450 px circle at 4 deg/s, now 8.4 px and 100 %. The gimbal model applies rate saturation (5 to 10 deg/s), acceleration saturation and a one-frame latency, and reports saturation to the log.

## 5 AI methods

The classical detector is fast and accurate in clear conditions but fails where "the brightest compact blob" is the wrong answer. A small fully convolutional network then provides the measurement. It runs only when the classical confidence is below a floor, and only on a $128 \times 128$ region around the prediction, so speed and reliability never depend on the model. It is a gap filler: in the standard scenarios it supplies 0 % of the measurements, and on a real 60 fps phone video about half.

**Network and training.** A three-level U-Net-style encoder with skip connections [10], about 0.1 M parameters, takes a normalised $128 \times 128$ patch and outputs a $64 \times 64$ heat map with a Gaussian bump on the beacon; the loss is weighted binary cross-entropy. Training data are rendered by the simulator itself under randomised backgrounds, beacon shapes and sizes, decoys, atmosphere presets and noise; the simulator knows the beacon position, so labels are exact and unlimited. The model is exported to ONNX and run with ONNX Runtime [11] on the CPU in a few milliseconds; sub-pixel position comes from a soft-argmax on the heat map followed by the same Gaussian refinement as the classical path. It is not used while a faint track is active: it was trained on visible beacons, and on a three-sigma patch its peak was often noise.

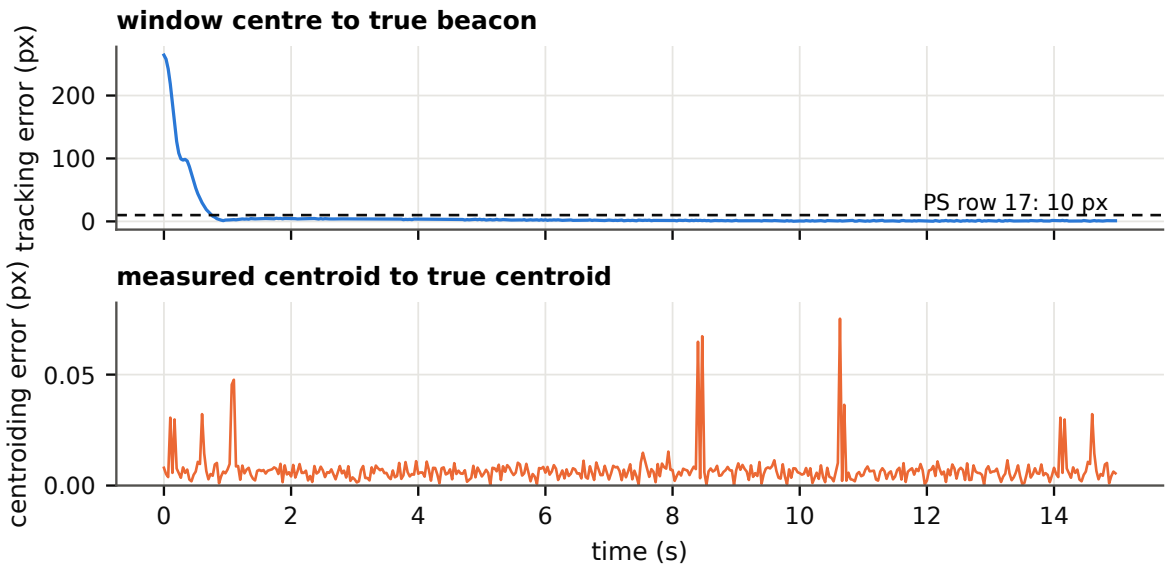



**Figure 4:** Both error terms on a clear circular path (15 s, seed 0, from the regression batch): the window settles within about a second and holds a few pixels, while the measured centroid stays within a few hundredths of a pixel of the truth.

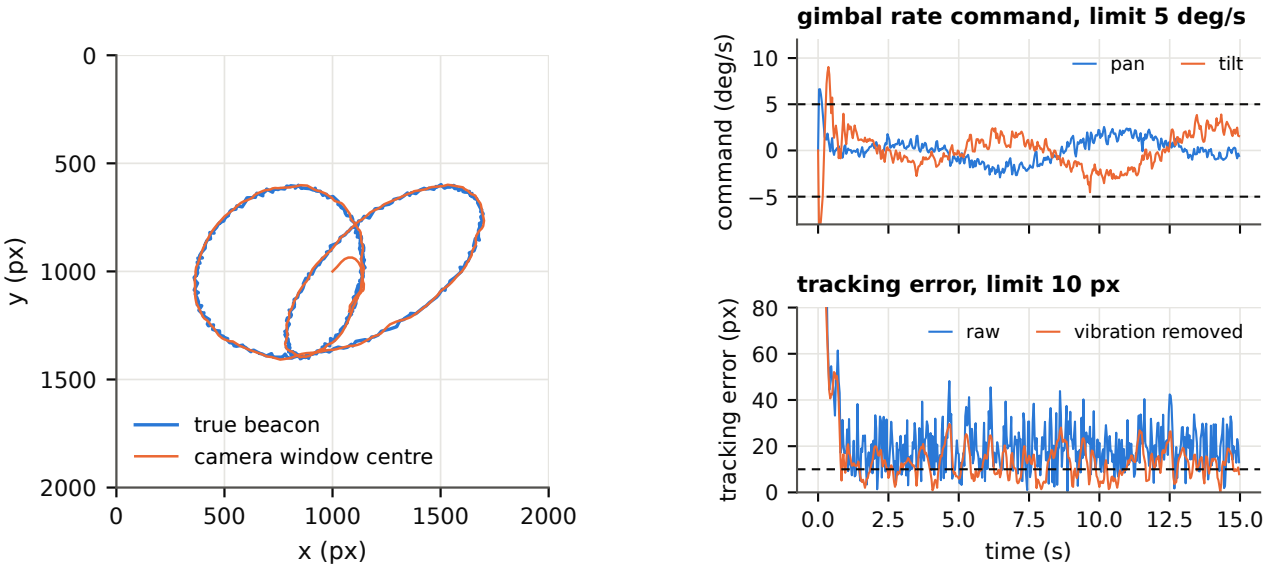



**Figure 5:** Left: true beacon path and camera window centre in the multi-target stress scenario (haze, salt and pepper, Gaussian and Poisson noise, circular sway, vibration, two decoys); the window follows the designated figure of eight throughout. Right: gimbal rate command against its 5 deg/s limit under platform sway plus vibration, and the tracking error raw and with the per-frame vibration removed.

## 6 The application

The deliverable is a standalone desktop application (PyQt6, packaged with PyInstaller for Windows, Linux, macOS Intel and macOS Apple silicon) that runs offline with no account. Its window has a parameter panel with one control per PS row and a tooltip naming that row; six live tiles that turn green when the specification is met (state, acquisition time, mean tracking error, mean centroiding error, lock retention, FPS); a scene view of the whole screen with the camera window, trails, a 2° grid and a legend; a camera view of what the window sees with the capture ring, the pointing-error vector, the detection box, the prediction with its uncertainty ring and a 1° scale bar; four live plots; and a telemetry column. Before a run the scene view previews $t = 0$ with every target named. A scenario picker, playback speed control and keyboard shortcuts support the demonstration. Disturbances can be changed while a run is going, as the “shall” item on introducing disturbances describes: a change

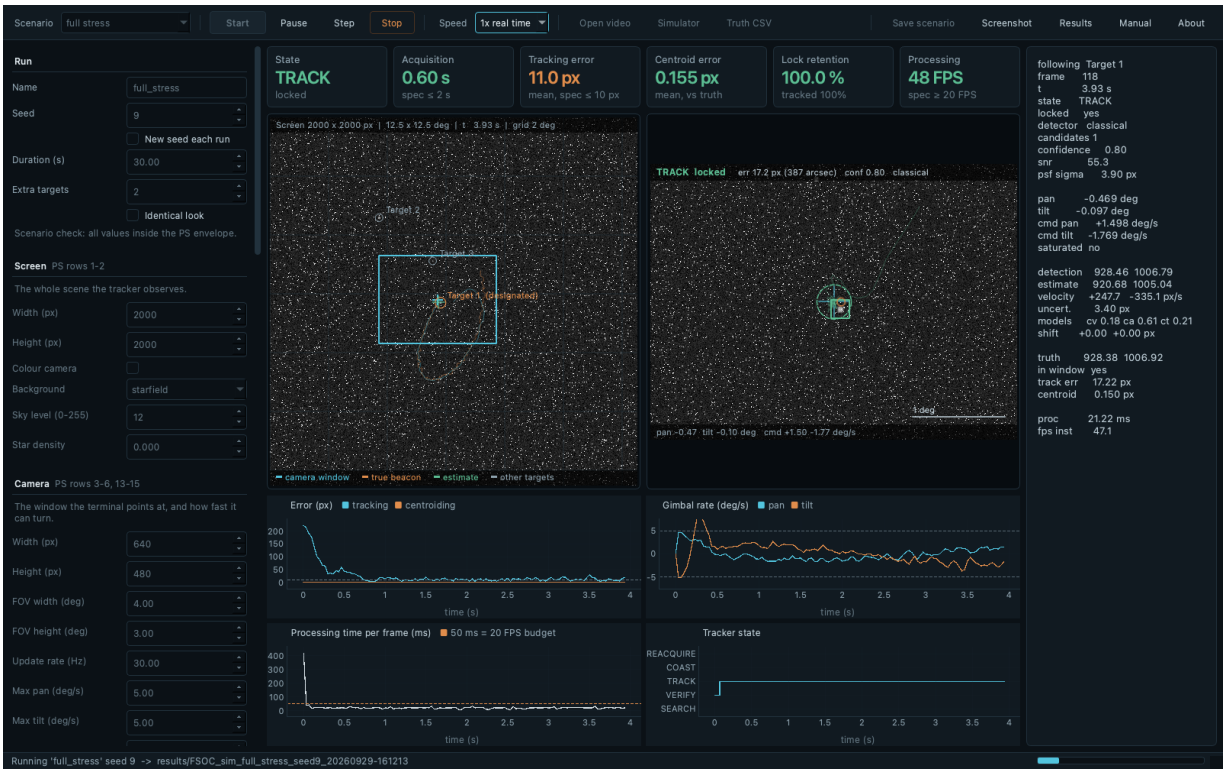



**Figure 6:** The desktop application locked on the designated beacon in the multi-target stress scenario: the parameter panel (left), the six specification tiles, the scene view with the camera window and the camera view with the capture ring, the four live plots and the telemetry column.

reaches the next frame, the tiles restart from it and the plots and report mark it; a scenario file can script the same changes.

"Open video" bypasses the simulator for Benchmark 2: the file's real size, frame rate (checked against the timestamps) and frame count calibrate the settings, the first frame is previewed, and the beacons found in the first 30 frames are listed for the operator to designate. "Truth CSV" loads a ground-truth file; without one the error tiles read "n/a" and lock is judged from the tracker's own estimate, which the report says. Every run ends with a dialog summarising the specification check and opens the PDF report on request. The web application at the project site presents the same panel, tiles and plots from the shared `ui_shared.py`; it is an addition, never a dependency of the desktop application.

# 7 Test methodology

- **Unit and closed-loop tests** (129, `pytest`): geometry and gimbal limits; determinism and screen bounds of every motion type; centroid accuracy on clean frames; IMM prediction; closed-loop specification checks on a clear scenario and under platform motion with vibration; a Benchmark 2 path that writes a synthetic video and runs the tracker on it; designation among decoys and among a video's beacons; the fixes from an independent review and from the PS audit, each pinned by a test.
- **PS audit** (`tools/ps_audit.py`) measures every PS item by running the code: rows 1 to 25, the eight "shall" functions, the deliverables, Benchmark 1 and Benchmark 2 with a rendered noisy video and its truth. 39 of 39 pass; the result is committed as `docs/PS_AUDIT.md`.
- **Regression rule.** Any change to perception, estimation or control is run on the whole scenario pack (15 s, seeds 0 to 2) before and after and compared run by run; no run may be worse. Nothing is tuned to a single video or scenario.
- **Scenario pack** (`configs/scenarios/`, Appendix A): the four mandatory motions in clear conditions, heavy noise at *PS row 21, 22* levels, fog, low light, a faint beacon, platform sway with vibration, the PS maximum at 5 and 10 deg/s, a multi-target stress case, identical decoys, mixed

beacon shapes, a fast circle, hard mode and a coded beacon: 17 scenarios plus an evaluator template.

- **Baseline comparison** (`tools/compare_trackers.py`): a deliberately simple tracker (brightest spot, proportional control) through the same simulator, gimbal and metrics, to show what each part of the design adds.
- **Reproducible figures.** `fsoc-tracker verify <run>` rebuilds every metric of a run from its per-frame CSV alone and checks the summary, so a reader can confirm any reported number. Every metric has a printed definition (Appendix B). Identical tracking numbers were obtained on Windows, Linux and both macOS builds.

# 8 Performance analysis

All figures in this section are measured by the software on the regression batch that ships with the report (`fsoc-tracker batch`, 15 s per run, seeds 0 to 2, laptop CPU); the charts are drawn from that batch's `envelope.json` by `docs/report/make_charts.py`, so a printed number and a plotted one are the same number. Ranges run over the three seeds. Tracking error is the mean distance from the true beacon to the window centre after acquisition; centroiding error the mean distance from the measured to the true centroid; lock retention the share of frames after acquisition in TRACK with the estimate within 30 px of the window centre; FPS is frames over processing time.

**Table 2:** Per-scenario results over seeds 0 to 2 (minimum to maximum; one value when the seeds agree). Heavy noise is salt and pepper 10 %, Gaussian $\sigma = 20$ and Poisson together. PS limits: acquisition $\leq 2$ s, tracking error $\leq 10$ px, lock $\geq 95$ %, FPS $\geq 20$.

| Scenario | Acquisition (s) | Tracking err. (px) | Centroiding (px) | Lock (%) | FPS |
|---|---|---|---|---|---|
| clear line | 0.70 to 1.07 | 2.4 to 3.4 | 0.006 | 100.0 | 201 to 213 |
| clear circular | 0.60 to 1.33 | 2.1 to 2.3 | 0.006 to 0.007 | 100.0 | 195 to 212 |
| clear figure 8 | 0.70 to 1.23 | 2.5 to 3.7 | 0.006 to 0.007 | 100.0 | 199 to 218 |
| clear random | 0.67 to 1.03 | 5.1 to 6.3 | 0.006 to 0.007 | 100.0 | 189 to 209 |
| heavy noise | 0.67 to 1.00 | 2.4 to 3.7 | 0.186 to 0.191 | 100.0 | 104 to 109 |
| fog | 0.63 to 1.33 | 2.5 to 3.0 | 0.078 to 0.080 | 100.0 | 118 to 124 |
| low light | 0.87 to 1.57 | 3.1 to 3.7 | 0.139 to 0.149 | 100.0 | 116 to 117 |
| faint beacon | 1.03 to 2.00 | 2.4 to 3.4 | 0.325 to 3.954 | 95.9 to 97.8 | 105 to 109 |
| fast circle 4 deg/s | 0.67 to 0.70 | 4.6 to 5.3 | 0.006 | 100.0 | 199 to 210 |
| identical decoys | 0.60 to 0.63 | 3.3 to 3.5 | 0.025 to 0.027 | 100.0 | 94 to 96 |
| beacon shapes | 0.67 | 2.6 to 3.1 | 0.041 to 0.061 | 100.0 | 86 to 95 |
| coded beacon | 0.97 to 1.87 | 1.8 to 2.9 | 0.028 to 60.666 | 100.0 | 87 to 96 |
| full stress | 0.30 to 1.03 | 10.3 to 11.3 | 0.160 to 0.232 | 98.2 to 100.0 | 114 to 118 |
| sway 12 + shake 20 | 0.70 to 0.80 | 19.0 to 19.4 | 0.020 to 0.021 | 99.3 to 100.0 | 76 to 86 |
| platform max 5 deg/s | 0.73 to 0.97 | 21.7 to 22.6 | 0.019 to 0.180 | 95.8 to 97.2 | 77 to 84 |
| platform max 10 deg/s | 0.80 to 0.97 | 22.6 to 23.5 | 0.019 to 0.180 | 93.4 to 94.1 | 77 to 85 |
| hard mode | 2.83 to 11.97 | 2.6 to 3.7 | 0.007 to 0.029 | 100.0 | 113 to 139 |

The coded beacon's centroiding range on two seeds includes the frames before acquisition in which a look-alike was measured and turned down by the code; once in TRACK the beacon is followed 100 % of the time. The faint beacon's upper centroiding value is one seed at three sigma, where the fit itself is noisy.

**Discussion.**

- Centroiding accuracy is set by the sub-pixel Gaussian fit and holds at 0.006 to 0.007 px in clear air and about 0.2 px under the heaviest noise the PS lists. This is the quantity compared in both benchmark rounds.
- On smooth paths in clear, noisy, foggy and low-light conditions the tracker meets every limit of *PS row 16 to 20* with 100 % lock. The random walk stays the highest clear case because its accelerations are unpredictable by construction: the controller can only lead what the estimator

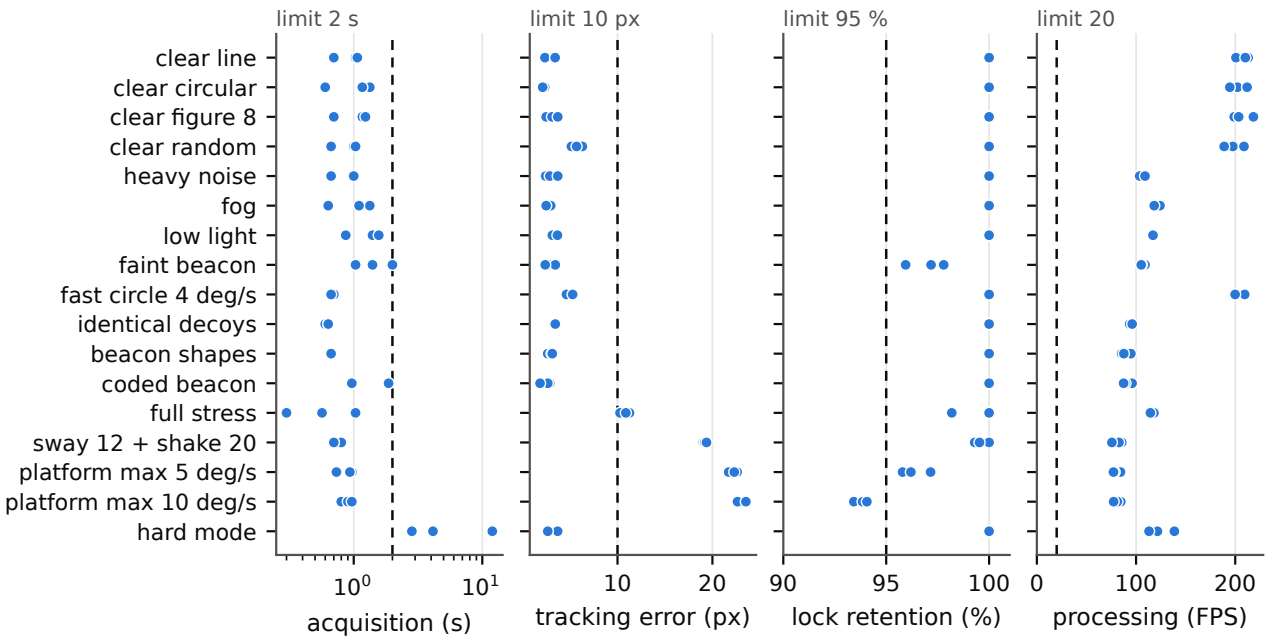



**Figure 7:** Every run of the batch against the four performance rows: acquisition time (*PS row 16*, log scale), mean tracking error (*PS row 17*), lock retention (*PS row 18*) and processing rate (*PS row 20*); one dot per seed, dashed lines at the PS limits. The runs beyond a limit are the platform-vibration settings, the full-stress case that stacks that vibration with noise, haze and decoys, and hard mode's acquisition, which is a blind sweep of the window by design.

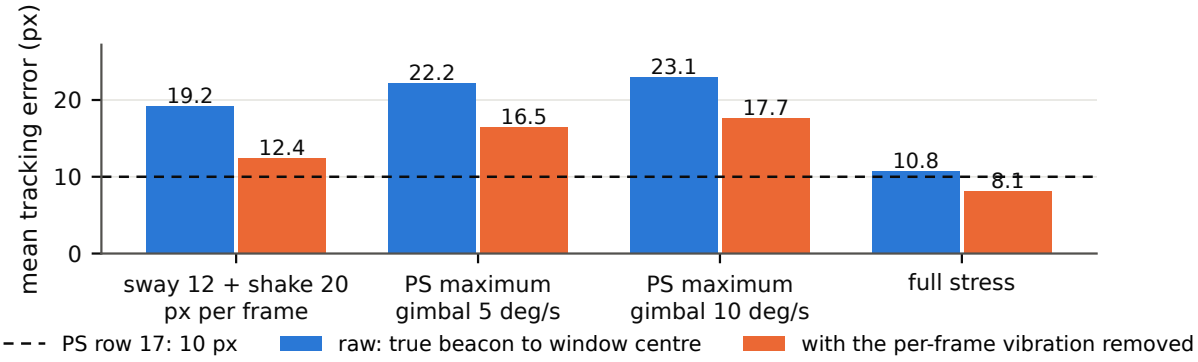



**Figure 8:** Tracking error raw and with the per-frame vibration removed. The removed part is unknowable before the frame arrives; the remainder is what a rate-limited gimbal can physically follow. Both are printed in every report.

can extrapolate.

- Under vibration the raw tracking error exceeds 10 px because the true position jumps up to 20 px every frame. At the PS maximum (20 px per frame sway plus 20 px per frame vibration) the binding limit was measured, not assumed: with the gimbal at the 10 deg/s the PS allows, slew saturation falls from about a third of frames to 2 % yet lock and error do not improve. The vibration is the limit, not the motor: a random jump is unknowable before the frame arrives, so each frame the pointing is off by roughly the jump. A wider field of view or a sensor larger than the window (electronic stabilisation) would remove it; neither is within the PS. This is documented, not tuned away.
- Identity: with identical look-alikes starting apart the start cue gives 100 % lock; look-alikes that start at the same point cannot be told apart at the start, which the scenario check states before the run. A coded beacon resolves look-alikes without any cue.
- The faint beacon (three to six sigma per frame) is acquired within the limit and held at 94 to 98 % over ten seeds; its hardest seeds sit just above the 5 % loss limit.
- Hard mode meets every limit but acquisition, which is a blind sweep of the window at the allowed

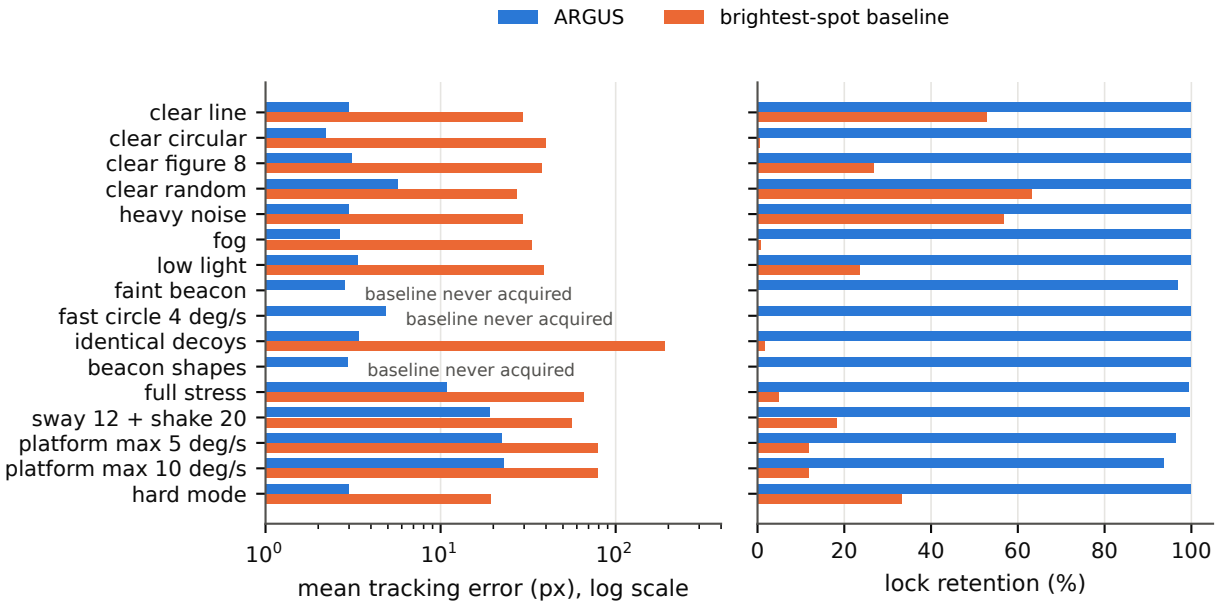



**Figure 9:** ARGUS against the brightest-spot baseline: mean tracking error (log scale) and lock retention per scenario, means over three seeds.

turn rate; by default the tracker observes the whole scene, the documented reading of the PS.

**Against a simple baseline.** Figure 9 compares ARGUS with the brightest-spot baseline through the same simulator and metrics (every scenario, seeds 0 to 2, 15 s). ARGUS meets every PS limit in 33 of 48 runs, the baseline in 0 of 48: its tracking error is 27 to 40 px on clear skies against 2 to 6 px, it holds 1.6 % lock among identical decoys and never acquires the faint beacon. It acquires faster on some clear skies because it has no confirmation step. The matched filter, the confirmation, the IMM lead, the gating and the identity signature each remove a failure the baseline shows.

**Benchmark 2.** On a noisy line clip rendered to `.mp4` at 30 fps with its truth file: tracking error 4.85 px, centroiding error 0.188 px. On a hand-held phone video with two beacons the operator chose the second from the list; acquisition 0.83 s, lock 99.4 %, 157 FPS over 1016 frames; the error tiles read “n/a” because the video carries no truth, and the measured centre of every frame is in the log for comparison with the evaluators’ positions.

## 9 Limitations and future improvements

- **Platform vibration at the PS maximum** is a physical limit of a 640 px window (section 8). Electronic stabilisation over a larger sensor, or a wide-field finder feeding the narrow-field camera (dual-sensor PAT), would remove it and are the natural next step.
- **Look-alikes starting at the same point** cannot be separated at the start. A lock-in detector that integrates the beacon code over many frames could extend identity by modulation to the faint limit, where a single frame cannot show a bit.
- **A static faint beacon** is not covered by track-before-detect, which finds dim things by their motion; the PS does not ask for one.
- **Learning.** Reinforcement-learned gain scheduling on top of the classical controller, and a learned denoiser for extreme scintillation, trained in the same simulator.
- **Hardware in the loop.** The `FrameSource` and gimbal interfaces already isolate the simulator, so a real camera and pan-tilt unit can be substituted.
- Switching the designated target mid-run and manual camera control were considered and not built; the PS does not ask for them, and they would change what the performance log scores.

## 10 Conclusion

ARGUS implements the eight "shall" functions and every row of the parameter table, and measures itself against *PS row 16 to 20* on every run. Its performance figures come from the software's own log and can be rebuilt from the per-frame data by anyone. Where the specification cannot be met, the limit is physical, measured at the settings the PS allows, and stated beside the number that shows it.

## References

[1] Y. Kaymak et al., "A survey on acquisition, tracking, and pointing mechanisms for mobile free-space optical communications," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1104–1123, 2018.

[2] M. A. Khalighi and M. Uysal, "Survey on free space optical communication: A communication theory perspective," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2231–2258, 2014.

[3] H. Kaushal and G. Kaddoum, "Optical communication in space: Challenges and mitigation techniques," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 57–96, 2017.

[4] R. Abdelfatah, N. Alshaer, and T. Ismail, "A review on pointing, acquisition, and tracking approaches in UAV-based FSO communication systems," *Opt. Quantum Electron.*, vol. 54, art. 571, 2022.

[5] L. C. Andrews and R. L. Phillips, *Laser Beam Propagation through Random Media*, 2nd ed. SPIE Press, 2005.

[6] H. A. P. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Trans. Autom. Control*, vol. 33, no. 8, pp. 780–783, 1988.

[7] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation.* Wiley, 2001.

[8] I. S. Reed, R. M. Gagliardi, and H. M. Shao, "Application of three-dimensional filtering to moving target detection," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-19, no. 6, pp. 898–905, 1983.

[9] S. Thomas et al., "Comparison of centroid computation algorithms in a Shack-Hartmann sensor," *Mon. Not. R. Astron. Soc.*, vol. 371, no. 1, pp. 323–336, 2006.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *MICCAI 2015*, LNCS 9351, pp. 234–241.

[11] ONNX Runtime, onnxruntime.ai; G. Bradski, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000.

[12] Smart India Hackathon 2026, Problem Statement 26169, Department of Space / ISRO, Space Applications Centre.

# A Scenario pack

| File | Purpose | Beacon | Disturbances |
|---|---|---|---|
| clear_line, clear_circular, clear_figure8, clear_random | the four mandatory motions (*PS row 12*) | square 10 px; line 150 px/s, circle radius 450 px, figure of 8 radius 500 px, random walk 140 px/s | none |
| noisy_line | *PS row 21, 22* at their stated levels | line | salt and pepper 10 %, Gaussian $\sigma = 20$, Poisson |
| fog_circular, lowlight_figure8 | *PS row 24* | circular; figure of 8 | fog preset, Gaussian 8; low-light preset, Gaussian 12, Poisson |
| lowlight_faint | beyond the single-frame detector | Gaussian 8 px, intensity 150 | low light, noise |
| platform_jitter | *PS row 23, 25* | circular, centred | linear sway 12 px/frame, vibration 20 px/frame |
| platform_max, platform_max_10degs | *PS row 23, 25* at the PS maximum, gimbal at 5 and 10 deg/s | circular, centred | linear sway 20 px/frame, vibration 20 px/frame |
| full_stress | *PS row 8, 21, 23, 24, 25* together | figure of 8 plus two decoys | haze, salt and pepper 5 %, Gaussian 12, Poisson, circular sway, vibration 10 |
| fast_circular | *PS row 13, 14, 17*: a fast target | circle radius 450 px at 4 deg/s (80 % of the turn rate) | none |
| hardmode_line | tracker restricted to the window | line | none |
| decoys_identical | *PS row 8*, designation by start | beacon plus three identical look-alikes starting apart, paths crossing | Gaussian 6, Poisson |
| beacon_shapes | *PS row 9, 10*, designation by appearance | $8 \times 18$ px rectangle among a cross, a ring, a diamond and a custom mask | none |
| coded_beacon | beyond the PS: identity by a blink code, no cue | the decoys_identical beacons; the designated one blinks 10110 at 10 bit/s | Gaussian 6, Poisson |

# B Performance log: metric definitions printed in every report

| Metric | Definition |
|---|---|
| Simulation duration | Last frame time minus first. |
| FPS | Frames over processing time (detection, estimation, control): 1000/mean processing ms. *PS row 20.* |
| Acquisition time | First frame in TRACK with the beacon within the capture radius of the window centre, from $t = 0$. *PS row 16.* |
| Tracking error, mean and maximum | Distance from the true beacon to the window centre over frames after acquisition. *PS row 17.* Also given with the per-frame vibration removed. |
| Centroiding error | Distance from the measured centroid to the true centroid over frames with a detection: mean, maximum, RMSE. |
| Tracked rate | Share of frames after acquisition in state TRACK regardless of centring; separates tracker failure from gimbal limits. |
| Lock retention | Share of frames after acquisition in TRACK with the beacon within the capture radius; target loss is 100 minus this. *PS row 18.* |
| Re-acquisition time | Time from losing lock to regaining it: count, mean, maximum. *PS row 19.* |
| Slew saturation | Share of frames in which the commanded rate exceeded the gimbal limit. |
| Handoff ready | Locked with the estimate within 10 px, held 1 s: where fine pointing could take over. |
| AI share | Share of frames in which the CNN provided the accepted measurement. |

Each run writes a folder `FSOC_<sim|video>_<name>_seed<N>_<date-time>/` holding `_frames.csv` (one row per frame), `_summary.json` (every metric with its definition and pass or fail), `_report.pdf` and `_scenario.yaml` (the exact parameters, so the run can be repeated).

# C Problem statement parameter table, row by row

| Row | Parameter | Suggested value | Implementation |
|---|---|---|---|
| 1 | Screen size | $\geq 2000 \times 2000$, user-defined | `ScreenConfig.width/height`, default $2000 \times 2000$ |
| 2 | Camera type | monochrome FPA, colour optional | monochrome default; colour renders three channels, tracker uses luminance |
| 3, 4 | Resolution, FOV | $640 \times 480$; $4^\circ \times 3^\circ$ | `CameraConfig`; IFOV derived |
| 5, 15 | Update rate, interval | 30 Hz min; $\geq 20$ Hz | commands every frame at 30 Hz |
| 6 | Initial camera position | centre of the screen | gimbal pose 0 at the screen centre |
| 7, 8 | Target type, number | beacon spot; 1 mandatory, multiple optional | rendered spot with PSF; named targets, one designated |
| 9, 10 | Shape, size | user-defined, square; 5 to 20 px | seven shapes and a custom mask; width and height separate |
| 11, 12 | Start, motion | random; at least four motions | random, centre or $x, y$; all six motions, waypoints and static |
| 13, 14 | Pan and tilt speed | 5 to 10 deg/s, default 5 | enforced in the gimbal model |
| 16 to 20 | Acquisition, tracking error, loss, re-acquisition, FPS | $\leq 2$ s, $\leq 10$ px, $< 5\,\%$, $\leq 1$ s, $\geq 20$ | measured every run, pass or fail printed |
| 21, 22 | Image noise, its deviation | salt and pepper about 10 %, Gaussian, Poisson; $\sigma \leq 20$ | three independent switches; salt and pepper shown in percent |
| 23 | Camera jitter | $\pm 20$ px per frame | `jitter_px`, applied to the picture |
| 24 | Atmosphere | clear, haze, fog, rain, low light | five presets plus editable contrast, brightness, blur, turbulence |
| 25 | Platform motion | $\pm 20$ px per frame; linear mandatory | five bounded sway patterns with the configured peak speed |