

Ocean Voyager

Software Architecture, Navigation Algorithms & Runtime Options

Implementation reference for the powered pool autonomy stack • v45-centered, with evolution of earlier navigation schemes

Executive overview

The powered-pool software evolved from deterministic forward/turn sequences and visual wall-avoidance experiments into a concurrent, event-driven autonomy program. The current v45 controller separates physical motor execution, RGB acquisition, thermal acquisition, perception, mission arbitration, telemetry, and display-only pose estimation. The key architectural principle is that sensing continues while the motors execute physically meaningful-duration actions: the camera is not forced to stop merely because a turn or forward burst is in progress.

At mission level, v45 has three dominant behaviors: NORMAL stochastic exploration, COLLISION_RANDOM recovery, and TARGET pursuit. Wall contact has highest priority and can preempt target pursuit. A target may interrupt exploration only when active RGB and active thermal evidence are corroborated; pursuit is bounded and followed by a mandatory cooldown so ordinary exploration always resumes.

1. Runtime architecture

The implementation can be understood as four concurrent producers/executors around one main arbitration loop:

Component	Execution model	Responsibility	Shared interface
Main autonomy loop	Main thread	Consumes newest RGB frame, evaluates safety/RGB/thermal/fusion, advances mission state, writes telemetry.	LatestFrame, Motor, ThermalWorker, Pose
Motor	Dedicated daemon thread	Turns logical motor mode into GPIO activity. FORWARD uses calibrated sequential left/right pump timing; turns energize one side continuously.	Thread-safe mode + epoch
Camera	Dedicated daemon thread + rpicalm-vid subprocess	Reads MJPEG bytes, extracts JPEG frames, decodes, rotates 180°, publishes only the newest frame.	LatestFrame single-slot/condition
ThermalWorker	Dedicated daemon thread	Reads MLX90640, archives raw arrays/PNGs, analyzes blobs, keeps a short observation deque.	Thread-safe recent-observation buffer

1.1 Latest-frame camera design

The camera thread launches rpicalm-vid in MJPEG mode. It scans the byte stream for JPEG SOI/EOI markers, decodes complete images, rotates them 180° to match the installed camera orientation, timestamps them with a monotonic clock, and publishes them into LatestFrame. LatestFrame is intentionally a single-slot structure guarded by a threading.Condition: a new image replaces the old image rather than joining a backlog.

The main loop waits for a frame ID newer than the one it last processed and rejects frames older than --max-frame-age. This design was a direct response to pool experiments: old frames are more dangerous than skipped frames because the boat continues moving. Earlier testing showed roughly 10 fps could be sustained while motor actions lasted much longer than one frame interval.

1.2 Motor execution thread

Motor.set() changes a logical mode—OFF, FORWARD, TURN_LEFT or TURN_RIGHT—and logs an epoch. The daemon motor thread continuously implements that mode. TURN_LEFT and TURN_RIGHT energize one pump side continuously. FORWARD is not assumed to mean equal simultaneous motors: it executes the calibrated left duration, optional gap, right duration, and optional gap while repeatedly checking that the requested mode has not changed. A state transition can therefore interrupt a long forward cycle safely.

This is an important implementation detail: mission logic specifies intent; the motor thread owns the low-level timing. The main autonomy loop can continue processing fresh camera frames while the motor thread is still executing the physical maneuver.

1.3 Thermal worker

The MLX90640 worker is independent of RGB. It uses I²C at 400 kHz, normally attempts a 2 Hz sensor refresh, saves each valid 24×32 temperature frame as .npy and optionally as a colorized PNG, analyzes the frame, and stores recent candidate observations in a bounded deque. The main loop asks for a temporally recent thermal candidate rather than blocking on a thermal read.

1.4 Main-loop priority

The main loop is the arbiter. For each fresh RGB frame it updates display pose, computes wall metrics, periodically performs YOLO inference, obtains the best recent thermal candidate, evaluates fusion, and then applies behavior priority. Confirmed wall contact preempts everything. If there is no collision takeover, an active TARGET state is serviced next. Otherwise a fused target can take over from NORMAL exploration if cooldown has expired; if not, the current random action runs until its deadline.

2. Behavioral state machine

State	Entry	Behavior	Exit
RANDOM	Mission start; target ends; collision clears	Draw action from NORMAL bank and run until deadline while sensing continues.	New normal draw, fused takeover, or confirmed collision
COLLISION_RANDOM	Confirmed wall contact	Draw LEFT/RIGHT/PAUSE from collision bank; after each action inspect current view.	If wall remains, draw again; otherwise return to NORMAL
TARGET	Valid fused RGB+thermal candidate outside cooldown	Steer to center target or issue adaptive forward approach bursts.	Reached/near, target lost, timeout, or wall contact
Cooldown	Not a separate motor state; timestamp gate after any target approach ends	Continue and log detections, but suppress another target takeover.	Expires after --target-cooldown-sec

3. Navigation schemes explored

Scheme	Core algorithm	Strength	Observed limitation / role
Deterministic square / lawnmower	Preplanned straight segments and nominal 90° turns.	Easy to reason about and plot.	Required repeatable motion and localization that the physical boat did not yet have.

CLOSE/OPEN visual avoidance	Classify the center view as safe/open versus close to wall; turn when close.	Simple reactive logic.	Single-frame edge/scale thresholds overlapped strongly across distances.
Temporal approach detection	Use multiple fresh frames and trends rather than one threshold.	More robust to texture/viewpoint than binary classification.	Still depended on sufficient steering authority.
Patient same-direction recovery	Latch a turn direction instead of rapidly flipping LEFT/RIGHT.	Reduced oscillation.	Tiny pulses could still make little physical progress.
Continuous turn-until-open	Keep one pump active while camera continues to update.	Much stronger steering authority.	Could circle indefinitely if OPEN was never recognized.
Bounded continuous turn	Add min/max turn duration.	Prevented unlimited circles.	Could re-enter FORWARD before actually escaping the wall.
STUCK recovery	Remember failed direction, create clearance, scan, avoid immediate forward.	Explicitly modeled failed recovery.	More state/complexity; physical validation was incomplete in that branch.
Visual drift correction	Estimate image motion with optical flow and compensate for left/right drift.	Potential trim without absolute pose.	Useful as an aid, not reliable enough to make exploration depend on it.
Timed counter-steer	Periodically inject a configured opposite steering pulse during forward travel.	Simple empirical correction for persistent curvature.	Open-loop; requires calibration.
Stochastic exploration (v45)	Randomly choose bounded FORWARD/LEFT/RIGHT/PAUSE actions; separate collision bank.	Tolerant of imperfect localization and propulsion asymmetry; produces genuine exploration.	Coverage is not deterministic and must be evaluated statistically/post-run.

4. v45 stochastic navigation algorithm

Action selection. Let p_i be the configured probability/weight for action i . A uniform random draw $u \in [0,1)$ is mapped through the cumulative probabilities. The selected action is the first i for which:

$$u < \sum_{j \leq i} p_j$$

Once action i is selected, its duration is sampled from its configured interval:

$$T_i = T_{\min,i} + v (T_{\max,i} - T_{\min,i}), \quad v \in [0,1)$$

A fixed --random-seed makes the u and v sequence reproducible. Sensor events can still preempt the selected action.

Each random-action specification is encoded as probability,min_seconds,max_seconds. For example 0.70,2.0,12.0 means a 0.70 weight/probability for that action and, when selected, a continuous duration sampled within 2–12 seconds. The controller constructs separate NORMAL and COLLISION banks. A random draw selects an action according to the configured probabilities; a second random draw selects its duration. --random-seed can make the sequence reproducible for debugging.

The successful Run 04 configuration used NORMAL: FORWARD 0.70 for 2–12 s, LEFT 0.07 for 1–6 s, RIGHT 0.17 for 1–6 s, PAUSE 0.06 for 0.2–4 s. COLLISION: LEFT 0.15 for 0.5–6 s, RIGHT 0.45 for 1–6 s, PAUSE 0.40 for 0.5–3 s. The asymmetric weights were deliberate responses to the boat's observed directional bias.

5. Wall / collision algorithm

Every fresh RGB frame is reduced to safety metrics E (edge-density style measure) and S (scale/distance-transform style measure). A hard wall condition is formed from $E \leq \text{--collision-edge-max}$ and $S \geq \text{--collision-scale-min}$. The condition must

persist for `--collision-confirm-frames` before takeover. On confirmation the motors are turned OFF, any active target pursuit is aborted, and a `COLLISION_RANDOM` action is selected.

At the end of a collision action, the current view is checked again. If the hard wall signature remains, another collision-bank action is drawn. If it has cleared, the controller logs `COLLISION_CLEAR` and returns to the `NORMAL` bank. This deliberately treats collision handling as recovery rather than requiring perfect pre-contact avoidance.

6. Visual drift algorithm

Optical-flow decision rule. If the RANSAC affine transform between two accepted frames is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & dx \\ c & d & dy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

then dx is the estimated horizontal image translation. With threshold $\tau = \text{--visual-drift-threshold-px}$:

$$\begin{aligned} |dx| < \tau &\rightarrow \text{STRAIGHT} \\ dx > \tau &\rightarrow \text{LEFT drift} \\ dx < -\tau &\rightarrow \text{RIGHT drift} \end{aligned}$$

The classification is accepted only when at least `--visual-drift-min-features` survive tracking; repeated classifications can then be required before adaptive trim is changed.

VisualDrift uses the upper ~55% of consecutive RGB images. `goodFeaturesToTrack` finds up to 120 features; `calcOpticalFlowPyrLK` tracks them; `estimateAffinePartial2D` with RANSAC estimates image translation. The x translation dx is compared with `--visual-drift-threshold-px`. With sufficient tracked features, small $|dx|$ is `STRAIGHT`; otherwise the sign is mapped to `LEFT` or `RIGHT` drift. A short history supports confirmation before adaptive correction.

`--visual-drift-mode` monitor observes/logs without changing propulsion. `adaptive` allows bounded timing adjustment using `--visual-adapt-step-sec`, `--visual-adapt-max-sec` and `--visual-adapt-cooldown-sec`. The feature-count, window and pixel threshold options control how readily a drift estimate is accepted.

7. RGB object detection

Horizontal error and bearing. For image width W and detected box center x_c :

$$e = (x_c - W/2) / (W/2)$$

so $e \approx -1$ is far left, $e = 0$ is centered, and $e \approx +1$ is far right. The code converts this to an approximate bearing using horizontal FOV θ_{FOV} :

$$\theta_{\text{RGB}} = e \cdot (\theta_{\text{FOV}} / 2)$$

Apparent target size is represented by normalized box area:

$$A = [(x_2 - x_1)(y_2 - y_1)] / (W H)$$

This A value is later used as the primary proximity signal during approach.

Vision loads the configured Ultralytics YOLO model. Inference runs every `--vision-every-frames` frames rather than on every camera frame. Detections below `--vision-conf` are discarded by YOLO; implausibly large boxes are then rejected using maximum total area, width fraction and height fraction. The highest-confidence plausible box is retained.

The box center is converted into normalized horizontal error in $[-1, +1]$. A bearing estimate is obtained by mapping that horizontal position across the configured horizontal field of view. Apparent box area is stored as a fraction of image area and later becomes the primary proximity signal for approach/reached decisions.

8. Thermal anomaly algorithm

Local thermal contrast. For temperature image $T(x,y)$, let $B(x,y)$ be the local median background. The anomaly score is:

$$\begin{aligned} D_{\text{warm}}(x,y) &= T(x,y) - B(x,y) \\ D_{\text{cold}}(x,y) &= B(x,y) - T(x,y) \end{aligned}$$

Pixels satisfying $D \geq \text{--thermal-threshold-c}$ are grouped into connected components. For a candidate blob C , the implemented quality score is approximately:

$$Q = \text{mean}(D \text{ over } C) \cdot \sqrt{|C|} \cdot P_{\text{edge}}$$

where $P_{\text{edge}} = 0.25$ for a component touching the frame edge and 1 otherwise. The blob with largest Q is retained.

Thermal centroid and bearing. Contrast values are used as weights w_k for pixels in the selected blob:

$$\begin{aligned} x_c &= \sum(w_k x_k) / \sum w_k \\ y_c &= \sum(w_k y_k) / \sum w_k \end{aligned}$$

For the 32-column MLX90640 frame, the horizontal bearing is then:

$$\theta_{\text{TH}} = [(x_c - 15.5) / 15.5] \cdot (\theta_{\text{FOV}} / 2)$$

The reported ΔT is the maximum local anomaly score within the selected blob.

For each 24×32 MLX90640 frame, the software constructs a local median background around each pixel. Depending on `--thermal-polarity`, it evaluates warm contrast (temperature minus local background), cold contrast (background minus temperature), or both. Pixels above `--thermal-threshold-c` form a binary anomaly mask. Four-connected components are extracted; components that are too small or implausibly large are rejected.

Each surviving blob receives a quality score based on mean contrast times the square root of blob size, with an edge penalty for blobs touching the thermal-frame boundary. The best blob is converted to a contrast $\Delta^\circ\text{C}$ and bearing using the thermal horizontal FOV. A candidate also needs temporal persistence: the configured number of recent thermal hits must fit inside `--thermal-window-sec` and their bearing spread must stay within `--thermal-bearing-spread-deg`.

9. RGB–thermal fusion

Fusion gate. In compact form, target takeover requires all of the following:

$$\begin{aligned} \text{FUSED} &= \text{RGB_active} \wedge \text{RGB_recent} \wedge \text{TH_active} \wedge (\Delta T \geq \Delta T_{\text{min}}) \\ &\quad \wedge (|\theta_{\text{RGB}} - \theta_{\text{TH}}| \leq \theta_{\text{agree}}) \wedge \text{cooldown_expired} \end{aligned}$$

where ΔT_{min} is `--thermal-min-delta-active-c` and θ_{agree} is `--sensor-bearing-agree-deg`. If the same evidence occurs during cooldown, it is logged but control takeover is suppressed.

Fusion is intentionally conservative. RGB must be active and satisfy its candidate-hit requirement; thermal must be active, have a current candidate, and exceed `--thermal-min-delta-active-c`. Both modalities must exist at the same decision point and their bearings must differ by no more than `--sensor-bearing-agree-deg`. Only then is `fused=True`.

If a fused candidate arrives during target cooldown it is still fully logged as FUSED_SUPPRESSED, but cannot take over navigation. Outside cooldown it causes a FUSED_CANDIDATE event, transitions the mission to TARGET, and initially commands OFF so target steering can take over cleanly.

10. Target steering and approach

Steering-duration scaling. Let e be normalized RGB horizontal error. When $|e|$ exceeds the deadband, the code computes

$$s = \min(1, |e| / e_{\text{hard}})$$
$$T_{\text{turn}} = T_{\text{min}} + (T_{\text{max}} - T_{\text{min}}) s$$

where $e_{\text{hard}} = \text{--target-hard-error}$ and $T_{\text{min}}/T_{\text{max}}$ are $\text{--target-turn-min-sec}$ and $\text{--target-turn-max-sec}$. Thus a small offset produces a short correction while a large offset approaches the maximum steering pulse.

Adaptive forward-burst equation. Let A be current normalized RGB box area and $A_{\text{near}} = \text{--approach-area-near}$. First compute

$$f = \text{clamp}(A / A_{\text{near}}, 0, 1)$$
$$T_{\text{base}} = T_{\text{max}} - (T_{\text{max}} - T_{\text{min}}) f$$

Then estimate positive recent area growth

$$g = \max(0, A_t - A_{(t-1)})$$

and shorten the burst using

$$T_{\text{approach}} = \text{clamp}[T_{\text{base}} / (1 + K \cdot 100g), T_{\text{min}}, T_{\text{max}}]$$

where $K = \text{--approach-growth-gain}$. The boat therefore advances more cautiously both when the target already looks large and when it is rapidly getting larger.

Reached / near-target rule. A frame contributes to the reached counter when

$$A \geq \text{--done-area} \quad \text{and} \quad |e| \leq \text{--done-center-error}$$

Repeated qualifying observations are required by --done-hits . Independently, $\text{--object-avoid-area}$ can terminate approach earlier to prevent driving physically into the object.

In TARGET, horizontal RGB error controls steering. If $|\text{error}|$ exceeds --target-deadbnd , the controller turns toward the target. Turn duration scales from $\text{--target-turn-min-sec}$ to $\text{--target-turn-max-sec}$ as error approaches $\text{--target-hard-error}$. If the target is centered, the controller commands a forward approach burst.

Approach duration is adaptive rather than fixed. It shrinks as apparent RGB area approaches $\text{--approach-area-near}$ and also shrinks when recent box area is growing quickly. This reduces the chance of charging into a rapidly approaching object. If RGB area reaches $\text{--object-avoid-area}$, the approach ends and the system performs a stochastic escape/cooldown behavior.

Target handling is bounded by $\text{--target-pursuit-max-sec}$ and --target-lost-sec . Reached logic uses RGB apparent area and centering over repeated hits; thermal is supporting evidence rather than the primary distance proxy. Any end to target approach starts the mandatory cooldown.

11. Pose estimation and renderer relationship

Display dead-reckoning equations. If estimated forward speed is v , heading is ψ ($0^\circ = +Y$), and Δt is elapsed time during FORWARD, the display pose integrates approximately as

$$\begin{aligned}x_{-}(t+1) &= x_{-}t + v \Delta t \sin \psi \\y_{-}(t+1) &= y_{-}t + v \Delta t \cos \psi\end{aligned}$$

During turns, heading is integrated using the configured display turn rate ω :

$$\psi_{-}(t+1) = \psi_{-}t \pm \omega \Delta t$$

These equations are for visualization/replay only. Wall-anchor corrections can reset the corresponding coordinate near a known pool boundary after a wall event; they do not convert the estimate into measured ground truth.

The real-time Pose object is explicitly display/replay oriented, not a claim of ground-truth localization. It integrates motor mode using `--display-forward-ft-s`, `--display-left-deg-s` and `--display-right-deg-s` and can apply wall-anchor corrections with `--wall-clearance-ft`. For the final Run 04 renderer the chosen external convention was $x=0-5$ ft, $y=0-7$ ft, start (2.5,0), heading +Y. The renderer reconstructs a plausible continuous route from logged actions and sensor events.

12. Telemetry and auditability

Each run creates an output directory with RGB observations, archived raw RGB frames, thermal .npy arrays, optional thermal PNGs, event logs, a line-buffered telemetry CSV, and a JSON snapshot of the run configuration including the effective random seed and parsed action banks. Telemetry includes time, frame, state, motor mode, estimated pose, E/S safety metrics, hard-wall count, random bank/action/duration, RGB confidence/error/area, thermal bearing/contrast, fused flag, cooldown remaining, and event annotation.

`--archive-rgb-every-frames` is especially important for offline localization experiments: it preserves raw camera evidence independently of the lower-rate annotated/status image saving. This is what makes it possible to try improved trajectory reconstruction later without rerunning the pool experiment.

13. Runtime option reference

The following tables describe the options exposed by the current v45 script. Some options remain for compatibility with earlier navigation schemes even when the v45 stochastic path is the primary mode.

13.1 Motors, stochastic navigation and trim

Option	Default	Meaning
<code>--forward-left-sec</code>	1.7 s	Left-pump ON duration within one calibrated FORWARD cycle.
<code>--forward-right-sec</code>	2.0 s	Right-pump ON duration within one calibrated FORWARD cycle.
<code>--forward-gap-sec</code>	0	Optional OFF gap between sequential pump bursts.
<code>--start-delay-sec</code>	0	Delay before mission begins.
<code>--normal-forward / left / right / pause</code>	see script	NORMAL-bank probability,min,max specification.
<code>--collision-left / right / pause</code>	see script	COLLISION-bank probability,min,max specification.
<code>--random-seed</code>	None	Fixed seed reproduces random action choices/durations.
<code>--forward-schedule</code>	empty	Optional forward-time ranges that override left/right forward timing.

--counter-steer-mode	off	off or timed periodic counter-steering during search forward travel.
--counter-steer-direction	right	Direction of timed counter-steer.
--counter-steer-every-sec	4.0	Forward-travel interval between counter-steer pulses.
--counter-steer-pulse-sec	0.20	Default counter-steer pulse length.
--counter-steer-schedule	empty	Optional elapsed-forward-time ranges with different counter-steer pulse lengths.

13.2 Camera and drift

Option	Default	Meaning
--width / --height	640 / 480	RGB capture dimensions.
--fps	10	Requested RGB camera frame rate.
--max-frame-age	0.45 s	Reject a frame if it is too old when consumed.
--visual-drift-mode	off	off, monitor, or adaptive.
--visual-drift-window-sec	0.7	Time separation between frames used for drift estimation.
--visual-drift-min-features	12	Minimum successfully tracked features.
--visual-drift-threshold-px	3.0 px	Horizontal-motion magnitude below which motion is treated as straight.
--visual-drift-confirmations	3	History length/confirmation requirement for drift direction.
--visual-adapt-step-sec	0.05 s	Increment used by adaptive trim.
--visual-adapt-max-sec	0.10 s	Maximum adaptive timing offset.
--visual-adapt-cooldown-sec	2.0 s	Minimum interval between adaptive changes.
--archive-rgb-every-frames	2	Save one raw RGB frame every N frames; 1 saves every frame.

13.3 Legacy/alternate search sequence controls

Option	Default	Meaning
--sequences	F:24; ...	Semicolon-separated deterministic search sequences; F uses calibrated forward cycles, L/R timed turns, O off.
--sequence-max-turn-sec	1.0 s	Upper bound on a sequence turn.
--sequence-switch-on-cycle	true	Whether to switch sequence after a cycle/wall event.

13.4 Collision/recovery

Option	Default	Meaning
--collision-edge-max	0.002	Maximum E for hard-contact classification.
--collision-scale-min	1000	Minimum S for hard-contact classification.
--collision-confirm-frames	3	Consecutive hard frames required before collision takeover.
--recovery-pause-sec	0.35 s	Compatibility parameter for earlier bounded recovery.
--recovery-turn-sec	3.0 s	Compatibility turn duration for earlier recovery scheme.
--recovery-max-pulses	4	Compatibility cap on recovery pulses.
--open-confirm-frames	6	Frames required to confirm open view in earlier recovery logic.
--recovery-forward-sec	4.0 s	Forward duration after earlier-style recovery.

--default-recovery-turn	left	Fallback direction in earlier recovery scheme.
--------------------------------	------	--

13.5 Sensor modes and RGB

Option	Default	Meaning
--vision-mode	active	off=no RGB inference; monitor=observe/log; active=eligible for control/fusion.
--thermal-mode	monitor	off=no thermal worker; monitor=observe/log; active=eligible for fusion/takeover.
--model	best.pt	YOLO model path.
--vision-conf	0.25	YOLO inference/logging confidence floor.
--vision-strong-conf	0.35	Labels detections STRONG versus WEAK.
--vision-imgsz	640	YOLO inference image size.
--vision-every-frames	3	Run YOLO every N RGB frames.
--vision-candidate-hits	1	RGB hit requirement for active candidate trigger.
--vision-max-area	0.12	Reject boxes occupying more than this image fraction.
--vision-max-width-frac / height-frac	0.55 / 0.55	Reject implausibly wide/tall boxes.
--sensor-bearing-agree-deg	18°	Maximum RGB–thermal bearing difference for fusion.

13.6 Thermal

Option	Default	Meaning
--thermal-polarity	warm	Detect warm, cold, or both types of anomaly.
--thermal-threshold-c	0.8°C	Per-pixel local-background contrast threshold.
--thermal-min-blob-pixels	4	Minimum connected-component size.
--thermal-fov-x-deg	55°	Horizontal FOV used for thermal bearing mapping.
--thermal-rate-hz	2.0	Thermal acquisition target rate.
--thermal-candidate-hits	2	Recent consistent thermal observations required.
--thermal-window-sec	1.8 s	Age window for candidate observations.
--thermal-bearing-spread-deg	15°	Maximum bearing spread across required hits.
--thermal-save-png	true	Save viewable thermal PNG in addition to raw .npy.
--thermal-min-delta-active-c	2.5°C	Minimum contrast for thermal to participate in active takeover.

13.7 Target approach and termination

Option	Default	Meaning
--candidate-forward	true	Compatibility control for moving during unconfirmed candidate tracking.
--candidate-timeout-sec	3.0 s	Compatibility candidate timeout.
--target-deadband	0.14	Normalized horizontal error considered centered enough for forward approach.
--target-hard-error	0.35	Error scale at which steering approaches maximum turn duration.
--target-turn-min-sec / max-sec	0.16 / 0.55 s	Bounds on target steering pulse

		duration.
--approach-forward-min-sec / max-sec	0.30 / 1.60 s	Bounds on target forward burst.
--approach-area-near	0.07	Apparent-area scale used to shorten approach bursts.
--approach-growth-gain	4.0	How strongly rapid area growth shortens approach.
--target-observe-sec	0.18 s	Compatibility observation timing.
--target-lost-sec	1.5 s	No-recent-RGB interval before target is declared lost.
--reacquire-turn-sec / attempts	0.30 s / 6	Compatibility reacquisition controls.
--done-area	0.10	RGB image-area fraction used by reached rule.
--done-hits	3	Repeated reached observations required.
--done-thermal-delta-c	2.5°C	Supporting thermal evidence threshold.
--done-center-error	0.20	Target must also be sufficiently centered for reached.
--object-avoid-area	0.085	End approach/escape when target appears this large.
--target-cooldown-sec	30 s	Mandatory no-retakeover period after any target approach ends.
--target-pursuit-max-sec	6 s	Hard upper limit on one target diversion.
--target-done	resume	Compatibility option; continuous mission resumes search.

13.8 Display/replay and output

Option	Default	Meaning
--pool-short-ft / long-ft	5 / 7 ft	Dimensions used by display pose.
--display-forward-ft-s	0.175 ft/s	Display-only forward speed calibration.
--display-left-deg-s	30°/s	Display-only left-turn rate.
--display-right-deg-s	45°/s	Display-only right-turn rate.
--wall-anchor	true	Enable display-pose wall anchoring.
--wall-clearance-ft	0.15 ft	Inset used when anchoring pose to a wall.
--output	outputs/pool_v45_random	Run output directory.
--save-rgb-sec	2.0 s	Interval for annotated/status RGB image saving.
--status-sec	1.0 s	Console status-print interval.
--dry-run	false	Exercise logic without energizing motors.

14. Recommended mode combinations

Purpose	Key settings	Behavior
Navigation-only pool test	--vision-mode off --thermal-mode off	Random/collision navigation without object takeover.
Observe sensors without control	--vision-mode monitor --thermal-mode monitor	Log RGB/thermal evidence while navigation remains unaffected.
RGB audit, thermal active only for logging	--vision-mode monitor --thermal-mode monitor	Useful for checking model/camera alignment without target pursuit.
Full fused autonomy	--vision-mode active --thermal-mode active	Only corroborated RGB+thermal evidence can interrupt exploration.
Repeatable stochastic debug	add --random-seed N	Replays the same random action/duration sequence, subject to

No physical motors	add --dry-run	sensor-triggered preemption. Exercises sensing/state/logging paths while motor GPIO remains off.
--------------------	---------------	--

15. Run 04 reference configuration

The demonstrated Run 04 configuration kept both RGB and thermal active, used the trained best.pt model, NORMAL bank $F=0.70/2-12$ s, $L=0.07/1-6$ s, $R=0.17/1-6$ s, $P=0.06/0.2-4$ s, COLLISION bank $L=0.15/0.5-6$ s, $R=0.45/1-6$ s, $P=0.40/0.5-3$ s, target pursuit capped at 5 s, target cooldown 30 s, and raw RGB archived every two frames. This configuration produced the successful demo with repeated RGB/thermal observations, fused candidates, target steering, cooldown suppression, collision recovery and continued stochastic exploration.

16. Software design lessons

- Separate sensing frequency from physical actuation duration.
- Prefer newest-frame semantics over queued camera frames for a moving robot.
- Keep low-level motor timing in its own executor so mission logic remains responsive.
- Make RGB and thermal independently observable and independently switchable between OFF, MONITOR and ACTIVE.
- Use temporal persistence and cross-sensor bearing agreement rather than letting one noisy observation seize control.
- Give safety/collision handling higher priority than target pursuit.
- Bound every diversion: target pursuit has a timeout and cooldown; random actions have deadlines.
- Treat physical asymmetry as a configurable calibration problem rather than hiding it in code constants.
- Log enough raw evidence to permit new trajectory/localization algorithms after the experiment.
- Keep display dead reckoning explicitly separate from claims of measured ground-truth localization.

Appendix A. Simplified control flow

CAMERA THREAD → newest RGB frame → MAIN LOOP

THERMAL THREAD → recent anomaly deque ┘

MAIN LOOP: stale check → pose update → wall metrics → optional YOLO → thermal candidate → fusion

if confirmed wall: COLLISION_RANDOM

elif TARGET: steer / approach / terminate

elif fused and cooldown expired: TARGET

elif random action deadline expired: draw NORMAL action

else: continue current action

MAIN LOOP → Motor.set(mode) → MOTOR THREAD → GPIO/pumps

MAIN LOOP → telemetry CSV + event log + RGB archive

THERMAL THREAD → .npy + optional PNG archive