Multi-Animal-Tracker

Technical Reference for the Current Tracking System

Detection, state estimation, assignment, identity continuity, ecosystem integration, and UI-to-runtime parameter mapping

Multi-Animal-Tracker PoseKit ClassKit DataSieve

This manuscript documents the current implementation of the tracking system in the repository as of March 4, 2026. It is meant to do four things at once:

- explain the algorithmic design to technically fluent readers,
- remain understandable to collaborators who are not tracking specialists,
- map runtime behavior back to the actual user interface controls,
- and situate tracking within the broader MAT ecosystem that also includes PoseKit, ClassKit, and DataSieve.

Primary implementation surfaces described in this reference include `multi_tracker.core.tracking.worker`, `multi_tracker.core.filters.kalman`, `multi_tracker.core.assigners.hungarian`, `multi_tracker.core.post.processing`, `multi_tracker.core.identity.runtime_api`, plus the associated GUI parameter derivation in `multi_tracker.gui.main_window`.

Contents

1 Executive Overview

Multi-Animal-Tracker (MAT) is not a single monolithic algorithm. The current system is a staged evidence-integration pipeline whose online and offline phases intentionally serve different purposes.

- The online phase answers: which detections belong to which tracks right now?
- The offline phase answers: after seeing the full video, which continuity claims are still defensible?

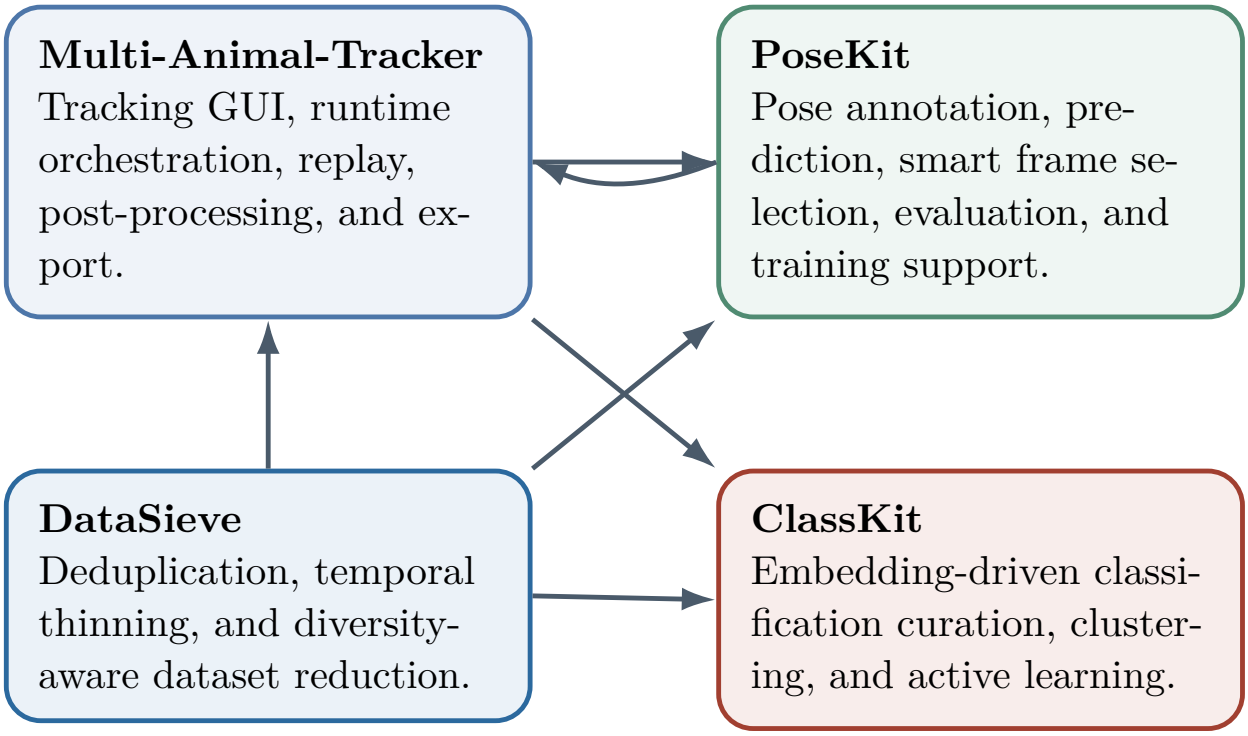
The present implementation therefore combines:

- per-frame object detection using either background subtraction or YOLO OBB,
- a vectorized Kalman state estimator with heading-aware process noise,
- weighted association over motion, orientation, geometry, and optional pose-shape cues,
- explicit lifecycle states for `active`, `occluded`, and `lost` tracks,
- an optional backward replay over the same cached detections,
- and a conservative post-processing stack for merge resolution, relinking, and interpolation.

The design bias is deliberate: when certainty is low, the system prefers fragmentation over a polished but wrong identity history.

2 Product Ecosystem

The tracking algorithm does not live alone. The repository is now a small ecosystem of tightly related tools.



Workflow summary. MAT exports crops and trajectory-derived datasets to PoseKit and ClassKit. PoseKit returns pose geometry and skeleton definitions used during tracking. DataSieve filters large image corpora before annotation, classification, or replay-oriented review.

Figure 1: Relationship between MAT and the adjacent tools in the repository.

2.1 Multi-Animal-Tracker

MAT

MAT is the tracking workbench. It handles:

- video ingest and preview,
- detection and tracking,
- backward replay,
- post-processing,
- export of CSV trajectories,
- and export of crops or frame subsets for downstream workflows.

2.2 PoseKit

PoseKit

PoseKit is the repository's pose-labeling and pose-inference application. From the current code it serves at least four roles:

- project-based keypoint annotation over image datasets,
- runtime-driven pose prediction using YOLO pose or SLEAP backends,
- embedding- and clustering-based smart frame selection,
- and downstream training / evaluation loops for pose datasets.

In practical workflow terms, MAT often feeds PoseKit by exporting individual crops or curated datasets for keypoint annotation. PoseKit then feeds back richer body geometry that MAT can use for directed heading and pose-aware association.

2.3 ClassKit

ClassKit

ClassKit is the image-classification sister framework. The current codebase positions it as an active learning dataset builder with:

- image ingestion into a SQLite-backed project store,
- embedding extraction through timm models,
- FAISS-based clustering of visual modes,
- uncertainty, diversity, representativeness, and audit sampling,
- and export / curation flows for image classification.

ClassKit is especially relevant when tracking outputs are later used to create behavior, morphology, or identity classification datasets.

2.4 DataSieve

DataSieve

DataSieve is the dataset subsampling and deduplication tool. Its current core includes:

- perceptual hash deduplication with pHash, dHash, and aHash,
- histogram-based similarity checks,
- color-signature comparisons,

- temporal subsampling,
- and clustering-based diversity sampling.

For large crop sets or frame corpora, DataSieve is the practical pre-filter before annotation in PoseKit or active learning in ClassKit.

3 Pipeline Topology

Figure 2 summarizes the current tracking flow.

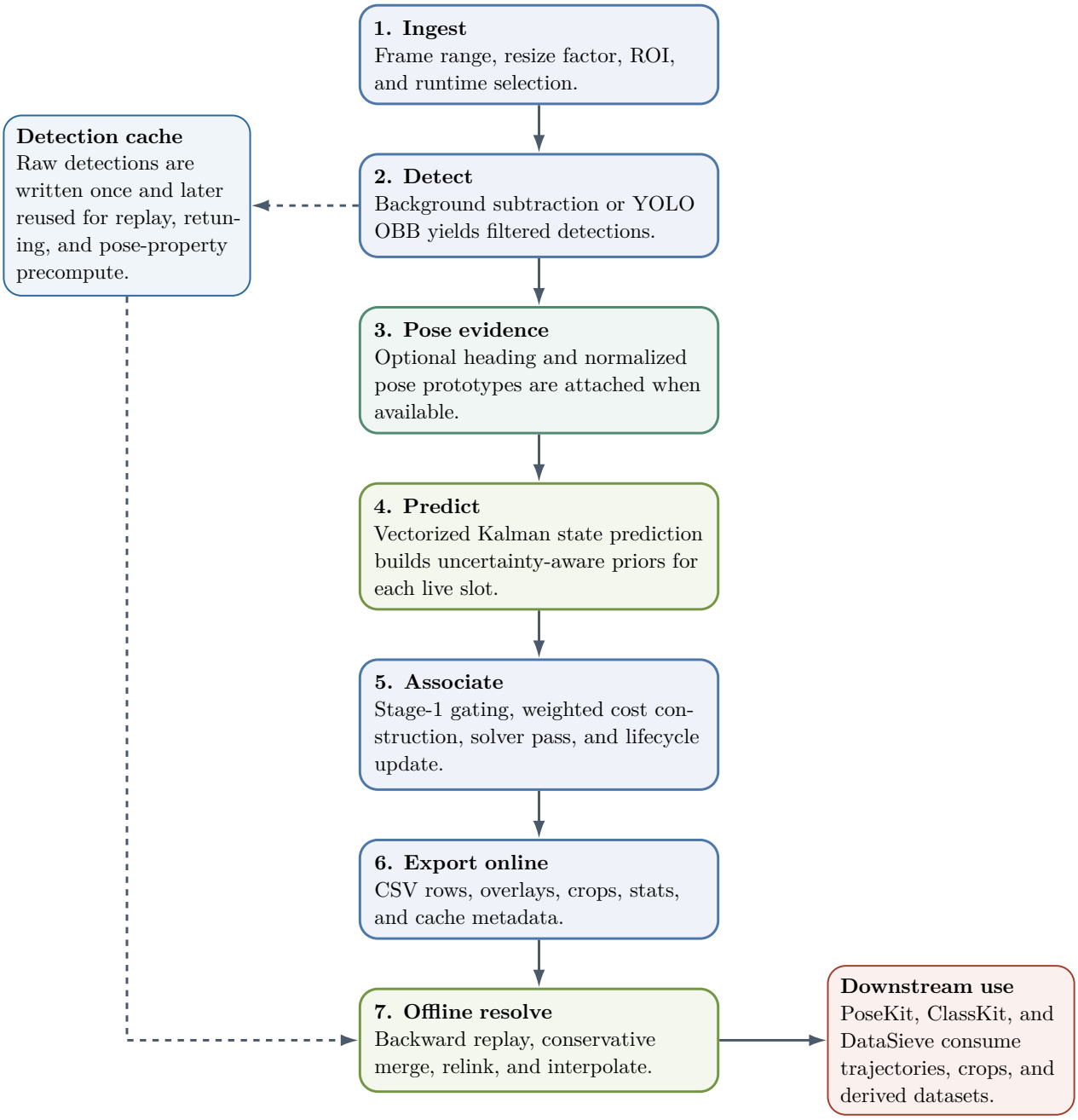


Figure 2: Operational flow of the current tracking pipeline.

The top-level worker performs the following broad stages:

1. read or replay the requested frame range,
2. detect candidate animals,
3. predict track states,
4. score and assign detections to tracks,

5. update lifecycle state and memory,
6. optionally run backward replay on the cached detections,
7. and finally resolve, relink, and interpolate trajectory outputs.

3.1 Per-frame execution order

The diagram above is the coarse view. At the level of one frame, the worker effectively executes the following inner loop:

1. acquire the resized frame and apply ROI / preprocessing decisions relevant to the chosen detector,
2. load cached detections or run the detector and then apply post-detection geometric filtering,
3. attach optional pose-side properties such as directed heading, keypoint visibility, and normalized pose prototypes,
4. predict every live Kalman slot in a vectorized batch,
5. construct candidate detection sets using cheap motion-and-geometry gates,
6. solve established-track assignment first, unstable-track assignment second, and guarded respawn last,
7. update Kalman state, lifecycle counters, heading memory, shape memory, and pose-feature EMAs,
8. write frame-level trajectory rows and any selected online exports,
9. and finally cache sufficient evidence for replay, post-processing, or later dataset extraction.

This ordering matters because the design intentionally spends cheap evidence first and expensive evidence later. Motion culling happens before detailed pose comparison; lifecycle transitions happen after assignment rather than before it; interpolation does not participate in the online identity decision at all.

The important architectural detail is that detection can be decoupled from tracking through the detection cache. That is what makes two-phase YOLO execution, backward replay, and reproducible tuning practical.

4 Formal Objects and Data Representations

4.1 State vector

Each track slot is modeled with the 5-dimensional state

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t & v_{x,t} & v_{y,t} \end{bmatrix}^\top . \tag{1}$$

The first two coordinates are center position in resized-frame coordinates. The angle θ_t is stored in radians. Velocity is measured in pixels per frame in the resized coordinate system.

4.2 Measurement vector

The online measurement update operates in the lower-dimensional space

$$\mathbf{z}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^\top. \tag{2}$$

4.3 Detection-side attributes

In the current implementation, association does not only consume $\mathbf{z}_t$. It also consumes parallel attributes:

- area,
- aspect ratio,
- confidence,
- OBB corners,
- deterministic `DetectionID`,
- optional pose-derived heading,
- optional normalized pose keypoints,
- optional pose visibility,
- and a crop-quality heuristic.

4.4 Deterministic detection identifiers

The tracker constructs IDs of the form

$$\texttt{DetectionID} = 10000 \cdot f + d, \tag{3}$$

where f is the absolute video frame index and d is the within-frame detection index. This simple encoding lets downstream tools recover frame identity from crop filenames and cache records.

5 Detection Layer

5.1 Background-subtraction branch

The background-subtraction branch is most useful in controlled scenes with stable backgrounds. The worker:

1. converts the frame to grayscale,
2. applies brightness, contrast, and gamma adjustments,

3. optionally performs lighting stabilization,
4. updates the adaptive background model,
5. generates a foreground mask,
6. applies ROI masking,
7. optionally applies conservative split morphology,
8. and extracts connected-component measurements.

This path is computationally cheap but scene-dependent.

5.2 YOLO OBB branch

The YOLO OBB branch emits oriented bounding boxes directly from the RGB/BGR frame. After inference, the system still performs downstream filtering, because inference and filtering are intentionally separated.

This separation matters operationally:

- the raw detections can be cached,
- ROI and size-filter choices can be changed without rerunning inference,
- and pose-property precompute can reuse the raw stream.

5.3 Detection cache

The detection cache is a first-class artifact in the current architecture. It is used for:

- forward replay without re-inference,
- backward replay,
- pose-property precompute,
- and reproducibility across parameter retuning.

Backward mode refuses to run without a compatible cache, which is a sensible integrity check rather than an implementation inconvenience.

5.4 Detection-to-track handoff

The detector does not hand the Kalman layer a bare centroid. The worker passes a richer detection record containing center position, orientation, area, aspect ratio, confidence, oriented corners, deterministic detection identity, and any optional pose descriptors.

That packaging choice is central to the current tracker design for two reasons:

- association can mix several weak cues rather than overcommitting to position alone,
- and offline stages such as merge resolution and relinking can reuse the same richer per-detection record without rerunning inference.

In other words, the current architecture is not just "detect then Kalman." It is "detect, annotate, cache, and only then decide identity."

6 State Estimation

6.1 Transition model

The current Kalman manager uses a constant-velocity-like state model with damping:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{w}_t, \tag{4}$$

with

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & d & 0 \\ 0 & 1 & 0 & 0 & d \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & d \end{bmatrix}, \tag{5}$$

where $d =$ `KALMAN_DAMPING`.

The system therefore allows velocity to persist, but only through a damped memory rather than an unconstrained ballistic extrapolation.

6.2 Observation model

Measurement projection is standard:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t, \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{6}$$

6.3 Heading-aware process noise

The process-noise model is one of the most important biological specializations in the current tracker. Longitudinal and lateral velocity noise are rotated by the current heading:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{7}$$

and the 2D velocity-noise block is

$$\mathbf{Q}_v(\theta) = \mathbf{R}(\theta) \begin{bmatrix} q_{\text{long}} & 0 \\ 0 & q_{\text{lat}} \end{bmatrix} \mathbf{R}(\theta)^\top. \tag{8}$$

The code realizes this directly in component form rather than building a dense symbolic rotation matrix. The practical interpretation is straightforward:

- uncertainty is larger along the likely direction of travel,
- uncertainty is smaller sideways,
- and the filter becomes less likely to overreact laterally during straight motion.

6.4 Prediction and correction

Prediction follows the standard form

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}\hat{\mathbf{x}}_{t-1|t-1}, \qquad \mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^{\top} + \mathbf{Q}_t. \tag{9}$$

Correction uses the innovation

$$\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_{t|t-1}, \tag{10}$$

with angle wrap logic applied to the orientation component so that $359^\circ \to 1^\circ$ is treated as a small change rather than a catastrophic one.

The innovation covariance is

$$\mathbf{S}_t = \mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^{\top} + \mathbf{R}, \tag{11}$$

and the gain is

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}^{\top}\mathbf{S}_t^{-1}. \tag{12}$$

The implementation uses the Joseph-form covariance update

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t|t-1}(\mathbf{I} - \mathbf{K}_t\mathbf{H})^{\top} + \mathbf{K}_t\mathbf{R}\mathbf{K}_t^{\top}, \tag{13}$$

which is more numerically stable than the naive covariance update.

6.5 Maturity and speed limits

The present implementation uses two operational safeguards beyond textbook Kalman logic:

1. **Age-dependent velocity retention.** Young tracks retain only a fraction of their predicted velocity until they reach `KALMAN_MATURITY_AGE`.
2. **Velocity clipping.** The corrected speed is clipped using `KALMAN_MAX_VELOCITY_MULTIPLIER` times `REFERENCE_BODY_SIZE`.

These two choices reduce explosive behavior from newly spawned or weakly supported tracks.

7 Orientation and Pose Handling

7.1 Directed versus axis-aligned orientation

The worker distinguishes between two orientation regimes:

- **Axis-aligned OBB orientation.** θ and $\theta + \pi$ represent the same body axis.
- **Directed pose-derived heading.** Pose keypoints identify an anterior and posterior side, so the angle is no longer ambiguous under a 180-degree flip.

7.2 Axis collapse

When pose does not provide a directed heading, the tracker resolves the OBB ambiguity by comparing

$$\theta_0 = \theta \bmod 2\pi, \qquad \theta_1 = (\theta + \pi) \bmod 2\pi, \tag{14}$$

and keeping whichever is closer to the previous reliable track orientation.

7.3 Pose-derived heading

If anterior and posterior keypoint groups are defined and visible above `POSE_MIN_KPT_CONF_VALID`, the worker computes centroids for both groups and forms

$$\theta_{\text{pose}} = \operatorname{atan2}(y_{\text{ant}} - y_{\text{post}},\, x_{\text{ant}} - x_{\text{post}}). \tag{15}$$

That direction is then normalized into $[0, 2\pi)$ and can override OBB orientation in the track update step.

7.4 Normalized pose prototypes

The worker also builds normalized pose prototypes by:

1. filtering invalid or ignored keypoints,
2. computing a weighted center,
3. centering the visible keypoints,
4. scaling by a robust median radius,
5. and storing a normalized (x, y, conf) array.

These prototypes are central to pose-aware association and later relinking.

8 Association and Online Identity Maintenance

8.1 Base positional uncertainty

For each active track, the worker derives an innovation covariance

$$\mathbf{S}_i = \mathbf{H}\mathbf{P}_i\mathbf{H}^\top + \mathbf{R}, \tag{16}$$

and the assigner caches $\mathbf{S}_i^{-1}$ for cost computation.

This matters because when `USE_MAHALANOBIS` is active, the positional term is

$$D_{ij}^{\text{pos}} = \sqrt{(\mathbf{m}_j - \mathbf{p}_i)^\top \mathbf{S}_{i,xy}^{-1}(\mathbf{m}_j - \mathbf{p}_i)}. \tag{17}$$

If Mahalanobis mode is disabled, the tracker falls back to ordinary Euclidean distance.

8.2 Orientation distance

For directed headings, the current orientation difference is the circular distance

$$D^{\text{ori-dir}}(a, b) = \min\left(|a - b|,\, 2\pi - |a - b|\right). \tag{18}$$

For axis-aligned headings, a 180-degree equivalence is allowed:

$$D^{\text{ori-axis}}(a, b) = \min\left(D^{\text{ori-dir}}(a, b),\, \pi - D^{\text{ori-dir}}(a, b)\right). \tag{19}$$

8.3 Stage-2 association cost

The online cost for track i and detection j is

$$C_{ij} = w_p D_{ij}^{\text{pos}} + w_o D_{ij}^{\text{ori}} + w_a \left|A_j - \bar{A}_i\right| + w_r \left|R_j - \bar{R}_i\right|, \tag{20}$$

where:

- A_j is detection area,
- $\bar{A}_i$ is the track's last remembered area,
- R_j is detection aspect ratio,
- $\bar{R}_i$ is the track's last remembered aspect ratio,
- and the weights are `W_POSITION`, `W_ORIENTATION`, `W_AREA`, and `W_ASPECT`.

8.4 Stage-1 gating

Before full advanced costing, the current implementation applies a track-specific candidate gate. Let

$$d_{\text{cull}} = \max\left(\frac{\texttt{MAX\_DISTANCE\_THRESHOLD}}{\max(\texttt{W\_POSITION}, \varepsilon)}, 50\right). \tag{21}$$

For track i, define

$$u_i = \min\left(2,\ \frac{\text{position uncertainty}_i}{\max(1, \texttt{REFERENCE\_BODY\_SIZE}^2)}\right),$$
$$q_i = \min\left(2,\ \frac{\text{avg step}_i}{\max(1, \texttt{REFERENCE\_BODY\_SIZE})}\right). \tag{22}$$

Then the local motion gate is

$$g_i = d_{\text{cull}} \cdot \texttt{ASSOCIATION\_STAGE1\_MOTION\_GATE\_MULTIPLIER} \cdot (1 + 0.5u_i + 0.35q_i). \tag{23}$$

A candidate pair is rejected before full scoring if any of the following hold:

- $D_{ij}^{\text{pos}} > g_i$,
- area ratio exceeds `ASSOCIATION_STAGE1_MAX_AREA_RATIO`,
- aspect-ratio change exceeds `ASSOCIATION_STAGE1_MAX_ASPECT_DIFF`.

8.5 Pose-shape rejection

If normalized pose prototypes exist for both the detection and the track, the assigner computes keypoint-wise distances on shared valid points. The current implementation then:

1. computes a median distance,
2. performs MAD-based outlier trimming,
3. trims the worst 20% if at least five distances remain,
4. and averages the retained distances.

The resulting scalar D_{ij}^{pose} is then used as a veto: if visibility is high enough and

$$D_{ij}^{\text{pose}} > \texttt{POSE\_REJECTION\_THRESHOLD}, \tag{24}$$

the candidate is discarded outright.

8.6 Assignment phases

The assigner uses three phases rather than a single matching call.

1. **Established tracks.** Tracks with continuity above `CONTINUITY_THRESHOLD` get first access to detections and are solved globally using Hungarian assignment or greedily if requested.
2. **Unstable tracks.** Lower-continuity tracks are greedily filled from the remaining detections.
3. **Lost-slot respawn.** Free detections may be assigned to lost slots if they lie far enough from all non-lost track predictions.

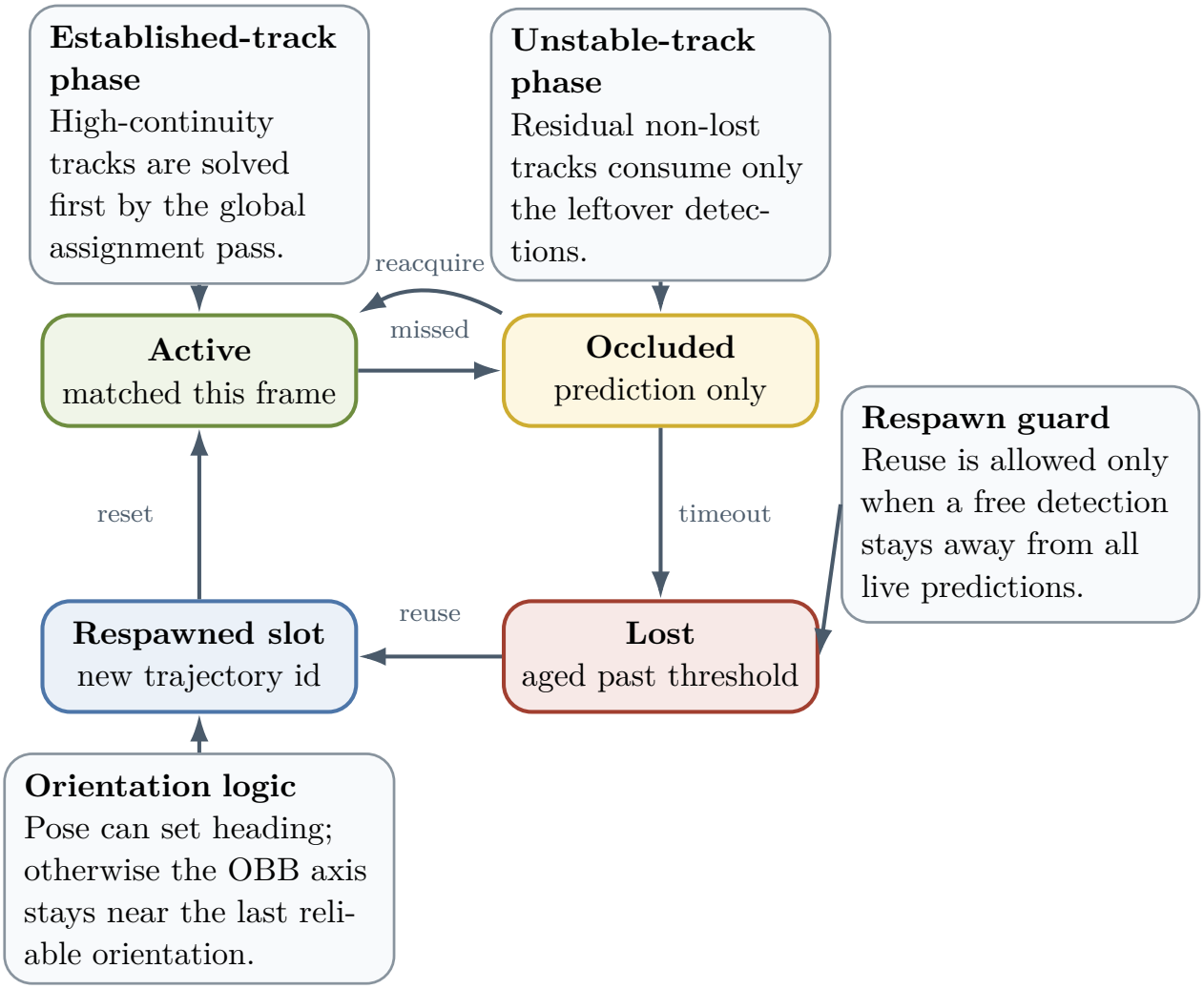


Figure 3: Association phases and lifecycle transitions in the current worker.

8.7 Respawn guard

Let $d_{j,\text{active}}$ be the minimum distance from free detection j to any non-lost predicted track position. Respawn is allowed only if

$$d_{j,\text{active}} \geq \texttt{MIN\_RESPAWN\_DISTANCE}. \tag{25}$$

This is a critical duplicate-suppression mechanism.

8.8 Track memory updates

The worker keeps more than Kalman state. It maintains:

- last trusted orientation,
- last shape information,

- a recent-position deque for speed estimation,
- an exponential moving average of track step size,
- and an exponential moving average of pose prototypes.

If the matched detection confidence is above `ASSOCIATION_HIGH_CONFIDENCE_THRESHOLD`, the average step is updated as

$$\bar{s}_i \leftarrow \alpha \bar{s}_i + (1 - \alpha) s_i, \tag{26}$$

with $\alpha =$ `TRACK_FEATURE_EMA_ALPHA`.

Pose prototypes are updated keypoint-wise using the same EMA logic when both track and detection have valid coordinates.

8.9 Why the worker prefers fragmentation

The worker's staged design makes a specific tradeoff: when two candidate identity explanations are similarly plausible, the system is biased toward splitting a trajectory rather than forcing a confident-looking but poorly supported continuation.

That bias appears in multiple places:

- stage-1 gating rejects impossible or weakly plausible pairs before the global solver sees them,
- pose rejection operates as a hard veto rather than a soft bonus,
- lost-slot reuse is guarded by explicit separation from live predictions,
- and forward/backward merge requires actual frame agreement rather than simple temporal adjacency.

For non-technical operators, this is the key interpretation rule: a fragmented result is often evidence that the tracker remained conservative under ambiguity, not evidence that the software "gave up."

8.10 Current limitations of pose and appearance cues

The present implementation can ingest pose-derived structure and can maintain lightweight appearance-adjacent memories such as shape, crop quality, and optional downstream identity artifacts. However, those cues are not yet reliable enough to serve as strong global assignment evidence inside the main Hungarian stage.

The limitation is empirical rather than philosophical. In the current tracker, pose and appearance fail for different reasons:

- **Pose is often incomplete at the exact frames where identity is hardest.** Crossings, overlaps, fast turns, motion blur, and partial occlusions are exactly the regimes that reduce keypoint visibility and destabilize anterior/posterior direction.

- **Appearance is weak for visually similar animals.** In many experiments, coat pattern, illumination, orientation, crop scale, and blur variation dominate any identity-specific visual signature available in a single short-term crop.
- **The assignment stage is high leverage and high risk.** A weak cue used inside a global solver can contaminate multiple pairings in one frame, which is more damaging than using the same cue later as a conservative veto or offline tie-breaker.

For that reason, the current system treats pose primarily as a direction cue and a rejection cue, not as a trusted identity embedding. Likewise, appearance-like signals are presently better viewed as downstream analysis inputs than as core online assignment evidence.

9 Backward Replay and Consensus Resolution

Backward replay is not a separate detector and not a separate learned model. It is a reverse-order pass over the same cached detections. Its value is epistemic: it provides a second trajectory hypothesis in scenes where crossings and occlusions create asymmetric ambiguity in forward time.

The current system therefore treats forward and backward outputs as peer hypotheses rather than forcing one to dominate the other.

9.1 Replay decision ladder

The replay stage is easiest to understand as a sequence of increasingly strict questions:

1. if the video were read in reverse, which detections would each identity claim?
2. where do the forward and backward fragments overlap in time?
3. on those shared frames, how often do both hypotheses place the same animal in nearly the same location?
4. if they disagree, is the disagreement localized enough that the safe action is to keep both fragments separate?
5. only after those tests pass, which fragments may be merged, stitched, or relinked?

This is why backward replay improves difficult scenes without acting like a magical correction step. It does not hallucinate new evidence; it only contributes a second interpretation of the same evidence stream.

10 Post-Processing and Trajectory Resolution

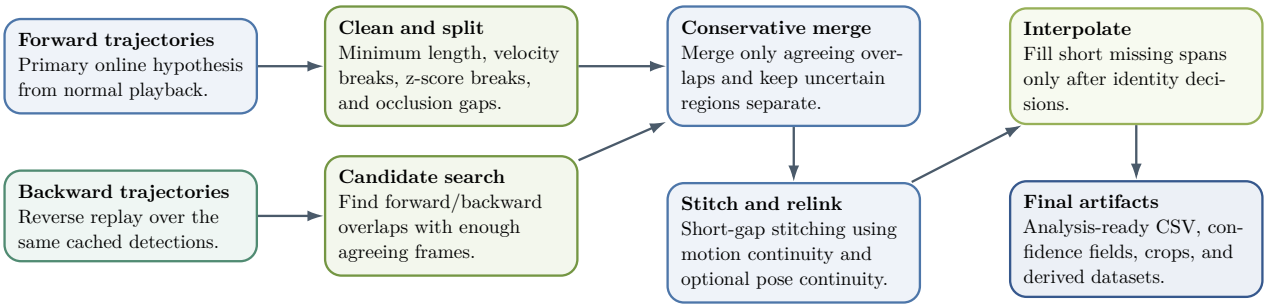


Figure 4: Current post-processing stack. The order is intentionally conservative.

10.1 Cleaning and breaks

The cleaning stage can:

- discard fragments shorter than `MIN_TRAJECTORY_LENGTH`,
- split at excessive absolute velocity using `MAX_VELOCITY_BREAK`,
- split at abnormal velocity z-scores,
- and split across long occlusion runs using `MAX_OCCLUSION_GAP`.

10.2 Forward/backward agreement

For a forward fragment F and backward fragment B, define the set of common frames

$$\Omega(F, B) = \{t \mid t \in F \cap B\}. \tag{27}$$

The agreement count is

$$A(F, B) = \sum_{t \in \Omega(F,B)} \mathbf{1}[\|\mathbf{p}_F(t) - \mathbf{p}_B(t)\|_2 \leq \texttt{AGREEMENT\_DISTANCE}]\,. \tag{28}$$

The pair is a merge candidate only if

$$A(F, B) \geq \texttt{MIN\_OVERLAP\_FRAMES}. \tag{29}$$

The implementation then performs conservative segment merging, redundancy removal, overlap-aware merge cleanup, and short-gap stitching.

10.3 Offline resolution ladder

The offline resolver can be read as a conservative ladder:

1. remove fragments that are too short or dynamically implausible,
2. search for forward/backward overlaps that agree spatially for enough frames,
3. merge only the overlaps that meet that agreement test,
4. relink nearby residual fragments only when motion and optional pose continuity remain credible,
5. and interpolate only after the identity boundaries have already been chosen.

That order is important. If interpolation or aggressive relinking were moved earlier, the tracker would be able to cosmetically hide identity uncertainty instead of exposing it.

10.4 Motion-and-pose relinking

The relinking stage summarizes each fragment by:

- start and end frame,
- start and end position,
- start and end heading,
- short-window velocity,
- and optional pose prototypes at the fragment ends.

For a source fragment a and destination fragment b, with temporal gap g, the algorithm predicts

$$\hat{\mathbf{p}}_{a\to b} = \mathbf{p}_a^{\text{end}} + \mathbf{v}_a^{\text{end}}(g+1), \tag{30}$$

and requires

$$\left\| \hat{\mathbf{p}}_{a\to b} - \mathbf{p}_b^{\text{start}} \right\|_2 \le \max(\texttt{AGREEMENT\_DISTANCE}, \texttt{MAX\_VELOCITY\_BREAK}(g+1)) \,. \tag{31}$$

If both fragments have informative motion, the heading difference must also satisfy

$$\Delta\theta \le \frac{\pi}{3}. \tag{32}$$

If both fragments have pose prototypes, the relinker also requires

$$D_{\text{relink}}^{\text{pose}} \le \texttt{RELINK\_POSE\_MAX\_DISTANCE}. \tag{33}$$

The candidate score used for greedy relinking is

$$\text{score}(a,b) = \frac{d_{\text{motion}}}{d_{\text{allowed}}} + 0.25\frac{g}{g_{\max}} + 0.35 h_{\text{norm}} + 0.75 p_{\text{norm}}, \tag{34}$$

where h_{norm} is normalized heading disagreement and p_{norm} is normalized pose disagreement.

10.5 Interpolation

Interpolation is the final polish stage, not the identity engine. The current system:

- reindexes each trajectory onto a complete frame range,
- fills missing state labels as `occluded`,
- interpolates x and y using the selected method,
- and interpolates θ with circular logic.

This prevents wrap-around artifacts around 0 and 2π.

11 Limitations and Future Directions

The current tracker is already strong as a conservative motion-and-geometry system, but the present implementation also makes its own ceiling visible. The strongest next gains are likely to come from better identity evidence and better post-experiment fusion, not from simply forcing today's weakest cues harder inside the online solver.

11.1 Why pose and appearance are not dominant Hungarian terms yet

One useful way to frame the current design is to distinguish the *implemented* online cost from the *tempting but not yet trustworthy* augmented cost. Today the main solver effectively works with

$$C_{ij}^{\text{online}} = C_{ij} + \lambda_{\text{pose-veto}} \mathbf{1}\left[D_{ij}^{\text{pose}} > \tau_{\text{pose}}\right], \tag{35}$$

where C_{ij} is the motion-and-geometry mixture from the online stage and the pose contribution behaves as a rejection rule rather than a soft identity attraction.

A more ambitious future formulation would look like

$$C_{ij}^{\text{future}} = C_{ij} + w_{\text{tag}} D_{ij}^{\text{tag}} + w_{\text{cls}} D_{ij}^{\text{class}} + w_{\text{emb}} D_{ij}^{\text{embed}}, \tag{36}$$

but the current system effectively runs with $w_{\text{cls}} \approx 0$ and $w_{\text{emb}} \approx 0$ in the online Hungarian stage because those signals are not yet robust enough across occlusion, blur, pose collapse, and visually similar animals.

This matters for three practical reasons:

- **Pose degrades at the worst moments.** Keypoints disappear or flip precisely during overlap, self-occlusion, and fast orientation change, which are also the frames where assignment has the highest leverage.
- **Appearance remains weak for many cohorts.** Lighting, viewpoint, crop scale, blur, and animal similarity often dominate any stable identity signature in a short crop.

- **Global assignment amplifies weak evidence.** A small but wrong identity preference in one frame's cost matrix can misallocate several tracks simultaneously.

That is why the current UI exposes pose mainly as conservative control points rather than as a large family of identity-embedding weights. In practice the operator-facing hooks are:

- `ENABLE_POSE_REJECTION`,
- `POSE_REJECTION_THRESHOLD`,
- `POSE_REJECTION_MIN_VISIBILITY`,
- and the heading-definition controls for anterior and posterior keypoints.

The system is conservative here on purpose.

11.2 Marker-mediated identity channels

The strongest near-term direction is explicit identity evidence that is externally readable and experimentally grounded rather than inferred from subtle visual similarity.

- **AprilTag-based identity.** When tags are visible and decoding is reliable, they provide a near-discrete identity observation.
- **Color-marker identity.** Deliberate color markers can play a similar role when printed tags are impractical for the assay geometry.

In a future marker-aware assigner, the safest use would be to treat tag or marker information first as a hard incompatibility gate and only second as a soft score bonus. One simple formulation is

$$D_{ij}^{\text{tag}} = \begin{cases} 0, & \text{marker agrees or is absent,} \\ +\infty, & \text{marker contradicts the track identity.} \end{cases} \tag{37}$$

That is a much better fit than simply increasing pose or appearance weights, because it upgrades the evidence quality itself. The UI already contains footholds for this direction:

- `COLOR_TAG_MODEL_PATH` and `COLOR_TAG_CONFIDENCE`,
- `APRILTAG_FAMILY` and `APRILTAG_DECIMATE`,
- plus identity-workflow controls such as `ENABLE_IDENTITY_ANALYSIS` and `IDENTITY_METHOD`.

11.3 Post-experiment identity fusion with contrastive learning

Another promising direction is post-experiment fusion using contrastive learning. The intended role is not to replace the online tracker with an embedding-only black box, but to operate *after* tracking when the full experiment and all exported crops are available.

One useful abstraction is to learn an embedding map

$$\phi(\mathcal{I}) \in \mathbb{R}^d \tag{38}$$

for crop or short-tracklet image sets $\mathcal{I}$, and then define an offline fragment affinity

$$S(a,b) = \alpha S_{\text{motion}}(a,b) + \beta S_{\text{time}}(a,b) + \gamma \cos\big(\phi(\mathcal{I}_a), \phi(\mathcal{I}_b)\big). \tag{39}$$

The current architecture is already a good host for this idea because MAT can export crops, ClassKit can curate identity-relevant image sets, and DataSieve can deduplicate or diversity-filter those sets before training. In practice:

- MAT remains conservative online,
- PoseKit and ClassKit help build structured post hoc training data,
- DataSieve trims redundant crop corpora,
- and the learned fusion model can operate only on residual fragments that survive the conservative motion pipeline.

11.4 Faster hybrid detection

Detection speed is another active direction. A practical next step is a two-stage detector that combines full-frame coarse localization with crop-level orientation refinement:

$$\mathcal{B}_{\text{coarse}} = f_{\text{YOLO-det}}(I_t), \tag{40}$$

$$\mathcal{O}_{\text{refined}} = \bigcup_{b \in \mathcal{B}_{\text{coarse}}} f_{\text{YOLO-OBB}}(I_t[b+\delta]), \tag{41}$$

where $I_t[b+\delta]$ denotes a padded crop around coarse box b.

The intended benefit is to run a fast detector once on the whole frame, spend OBB compute only on likely animal regions, preserve orientation-aware geometry for association, and keep the cache/replay architecture unchanged. This direction also maps well onto the current runtime surface: `YOLO_MODEL_PATH`, class filters, backend selection, TensorRT controls, and batching controls remain relevant, while likely extensions would add crop padding, coarse-detection thresholds, and refinement policy.

11.5 Current UI footholds for future work

Table 1: Future directions mapped to today's UI/runtime surface.

Future direction	Existing UI/runtime footholds	Expected next extension
Marker-mediated online identity	ENABLE_IDENTITY_ANALYSIS, IDENTITY_METHOD, COLOR_TAG_MODEL_PATH, COLOR_TAG_CONFIDENCE, APRILTAG_FAMILY, APRILTAG_DECIMATE	add marker-to-track compatibility gating and confidence-aware identity carry-over inside the assigner
Pose-informed but still conservative assignment	ENABLE_POSE_REJECTION, POSE_REJECTION_THRESHOLD, POSE_REJECTION_MIN_VISIBILITY, heading-definition controls	keep pose as a veto or directional cue until keypoint reliability supports a stronger identity term
Post-experiment contrastive fusion	crop export controls, ENABLE_INDIVIDUAL_DATASET, ENABLE_INDIVIDUAL_IMAGE_SAVE, relinking and merge controls	add an offline fragment-affinity model that scores residual fragments after tracking and before final export
Hybrid YOLO-detect + YOLO-OBB	YOLO_MODEL_PATH, runtime dropdowns, TensorRT and batching controls, class filters	add coarse-detector thresholding, crop padding, and crop-refinement scheduling without changing downstream association logic
Dataset curation for identity models	dataset generation controls plus DataSieve / ClassKit workflows	build tighter export loops for deduplication, hard-negative mining, and identity-balanced sampling

11.6 Longer-horizon roadmap

Beyond the immediate work above, the technical roadmap suggested by the current system is:

1. keep the online tracker conservative and interpretable,
2. add stronger explicit identity channels such as tags or color markers when the experiment permits them,
3. use richer learned fusion only after the experiment, where whole-trajectory evidence is available,
4. accelerate the detection front-end so the identity stack can spend compute where uncertainty is highest rather than everywhere uniformly,
5. and keep tightening the handoff among MAT, PoseKit, ClassKit, and DataSieve so downstream model-building workflows become part of the tracking loop rather than a manual afterthought.

This direction preserves the strongest property of the current tracker: it exposes uncertainty instead of hiding it. Future improvements should therefore increase evidence quality and speed without turning the core assignment stage into a black box that is hard to audit.

12 UI-to-Runtime Parameter Mapping

The GUI does not pass all values to the worker in raw user-entered form. Several controls are scaled into resized-coordinate pixel units. The mapping is implemented in `multi_tracker.gui.main_window: get\_parameters\_dict`.

12.1 Primary derived quantities

Let

$$s_{\text{ref}} = \texttt{REFERENCE\_BODY\_SIZE} \cdot \texttt{RESIZE\_FACTOR}, \tag{42}$$

and

$$a_{\text{ref}} = \pi \left(\frac{\texttt{REFERENCE\_BODY\_SIZE}}{2} \right)^2 \cdot \texttt{RESIZE\_FACTOR}^2. \tag{43}$$

Then the GUI derives several runtime values:

$$\texttt{MIN\_OBJECT\_SIZE} = \text{UI min size multiplier} \cdot a_{\text{ref}}, \tag{44}$$

$$\texttt{MAX\_OBJECT\_SIZE} = \text{UI max size multiplier} \cdot a_{\text{ref}}, \tag{45}$$

$$\texttt{MAX\_DISTANCE\_THRESHOLD} = \text{UI max assignment distance multiplier} \cdot s_{\text{ref}}, \tag{46}$$

$$\texttt{CONTINUITY\_THRESHOLD} = \text{UI recovery-search multiplier} \cdot s_{\text{ref}}, \tag{47}$$

$$\texttt{MIN\_RESPAWN\_DISTANCE} = \text{UI respawn multiplier} \cdot s_{\text{ref}}, \tag{48}$$

$$\texttt{VELOCITY\_THRESHOLD} = \text{UI velocity threshold} \cdot s_{\text{ref}}/\texttt{FPS}, \tag{49}$$

$$\texttt{MAX\_VELOCITY\_BREAK} = \text{UI max velocity break} \cdot s_{\text{ref}}/\texttt{FPS}, \tag{50}$$

$$\texttt{VELOCITY\_ZSCORE\_MIN\_VELOCITY} = \text{UI z-score min velocity} \cdot s_{\text{ref}}/\texttt{FPS}, \tag{51}$$

$$\texttt{AGREEMENT\_DISTANCE} = \text{UI merge agreement multiplier} \cdot s_{\text{ref}}. \tag{52}$$

12.2 Crosswalk tables

The following tables group runtime parameters by the user-facing UI areas that feed them.

Table 2: Reference scaling, frame range, and runtime fan-out.

UI area / control	**Runtime keys**	**Transformation**	**Primary consumers**
Video file + frame range	`START\_FRAME`, `END\_FRAME`	direct pass-through	`tracking.worker` frame iteration and cache coverage checks
Acquisition FPS	`FPS`	direct pass-through	velocity scaling, visualization, export interpretation
Reference body size	`REFERENCE\_BODY\_SIZE`	direct pass-through	scaling anchor for motion, geometry, crop quality, and relinking
Resize factor	`RESIZE\_FACTOR`	direct pass-through	frame resize, scaled body size, scaled area, visualization geometry

UI area / control	Runtime keys	Transformation	Primary consumers
Compute runtime dropdown	COMPUTE_RUNTIME, YOLO_DEVICE, ENABLE_TENSORRT, ENABLE_ONNX_RUNTIME, POSE_RUNTIME_FLAVOR, POSE_SLEAP_DEVICE	canonical runtime is translated into backend-specific fields	detection runtime, pose runtime, export behavior, UI option gating
TensorRT / batch controls	TENSORRT_MAX_BATCH_SIZE, ADVANCED_CONFIG.enable_yolo_batching, ADVANCED_CONFIG.yolo_batch_size_mode, ADVANCED_CONFIG.yolo_manual_batch_size	direct or derived from runtime requirements	two-phase YOLO batching and engine sizing

Table 3: Detection and preprocessing controls.

UI area / control	Runtime keys	Transformation	Primary consumers
Detection method selector	DETECTION_METHOD	enum pass-through	chooses background subtraction versus YOLO OBB branches
YOLO model + classes	YOLO_MODEL_PATH, YOLO_TARGET_CLASSES	model path resolution + optional class parsing	detector construction and inference filtering
YOLO confidence / IoU	YOLO_CONFIDENCE_THRESHOLD, YOLO_IOU_THRESHOLD, USE_CUSTOM_OBB_IOU_FILTERING	direct pass-through	YOLO raw filtering before association
Background threshold and morphology	THRESHOLD_VALUE, MORPH_KERNEL_SIZE, MIN_CONTOUR_AREA, MAX_CONTOUR_MULTIPLIER	direct pass-through	foreground segmentation and contour candidate extraction
Size filtering	ENABLE_SIZE_FILTERING, MIN_OBJECT_SIZE, MAX_OBJECT_SIZE	multipliers converted to scaled pixel area	background and YOLO post-filtering
Lighting stabilization	ENABLE_LIGHTING_STABILIZATION, LIGHTING_SMOOTH_FACTOR, LIGHTING_MEDIAN_WINDOW	direct pass-through	grayscale preprocessing in background mode
Adaptive background	BACKGROUND_PRIME_FRAMES, ENABLE_ADAPTIVE_BACKGROUND, BACKGROUND_LEARNING_RATE	direct pass-through	`core.background.model`

UI area / control	Runtime keys	Transformation	Primary consumers
Image adjustments	BRIGHTNESS, CONTRAST, GAMMA, DARK_ON_LIGHT_BACKGROUND	slider normalization for contrast and gamma	preprocessing helpers
Conservative split / dilation	ENABLE_CONSERVATIVE_SPLIT, MERGE_AREA_THRESHOLD, CONSERVATIVE_KERNEL_SIZE, CONSERVATIVE_ERODE_ITER, ENABLE_ADDITIONAL_DILATION, DILATION_ITERATIONS, DILATION_KERNEL_SIZE	direct pass-through	blob separation in crowded background-subtraction scenes
ROI editor	ROI_MASK	binary mask generated in UI and passed directly	detection masking and overlay drawing

Table 4: Tracking, Kalman, and association controls.

UI area / control	Runtime keys	Transformation	Primary consumers
Max targets	MAX_TARGETS	direct pass-through	Kalman slot count, trajectory buffers, color generation
Assignment distance multiplier	MAX_DISTANCE_THRESHOLD	multiplier $\times s_{\text{ref}}$	assignment acceptance ceiling and cull threshold derivation
Recovery search multiplier	CONTINUITY_THRESHOLD	multiplier $\times s_{\text{ref}}$	established-versus-unstable track split
Lost / respawn controls	LOST_THRESHOLD_FRAMES, MIN_RESPAWN_DISTANCE	respawn multiplier $\times s_{\text{ref}}$	lifecycle transitions and duplicate suppression
Start and stabilization counts	MIN_DETECTIONS_TO_START, MIN_DETECTION_COUNTS, MIN_TRACKING_COUNTS	direct pass-through	worker initialization and stabilization gating
Velocity threshold	VELOCITY_THRESHOLD	body-lengths per second $\rightarrow$ pixels per frame	orientation smoothing and flip logic
Orientation behavior	INSTANT_FLIP_ORIENTATION, MAX_ORIENT_DELTA_STOPPED	direct pass-through	tracking.worker:_smooth_orientation
Kalman process and measurement noise	KALMAN_NOISE_COVARIANCE, KALMAN_MEASUREMENT_NOISE_COVARIANCE	direct pass-through	Kalman Q and R magnitude

UI area / control	Runtime keys	Transformation	Primary consumers
Kalman damping and maturity	KALMAN_DAMPING, KALMAN_MATURITY_AGE, KALMAN_INITIAL_VELOCITY_RETENTION	direct pass-through	prediction damping and young-track restraint
Kalman velocity caps and anisotropy	KALMAN_MAX_VELOCITY_MULTIPLIER, KALMAN_LONGITUDINAL_NOISE_MULTIPLIER, KALMAN_LATERAL_NOISE_MULTIPLIER	direct pass-through	speed clipping and rotated process-noise design
Association weights	W_POSITION, W_ORIENTATION, W_AREA, W_ASPECT	direct pass-through	cost matrix assembly
Distance metric and solver	USE_MAHALANOBIS, ENABLE_GREEDY_ASSIGNMENT, ENABLE_SPATIAL_OPTIMIZATION	direct pass-through	cost metric and solver behavior
Stage-1 advanced gate	ASSOCIATION_STAGE1_MOTION_GATE_MULTIPLIER, ASSOCIATION_STAGE1_MAX_AREA_RATIO, ASSOCIATION_STAGE1_MAX_ASPECT_DIFF	direct pass-through	candidate pruning before advanced costing
Pose-aware assignment	ENABLE_POSE_REJECTION, POSE_REJECTION_THRESHOLD, POSE_REJECTION_MIN_VISIBILITY, TRACK_FEATURE_EMA_ALPHA, ASSOCIATION_HIGH_CONFIDENCE_THRESHOLD	direct pass-through	pose vetoing and track-memory EMA update

Table 5: Post-processing, merge resolution, and diagnostics controls.

UI area / control	Runtime keys	Transformation	Primary consumers
Enable post-processing	ENABLE_POSTPROCESSING	direct pass-through	post-run pipeline activation
Fragment cleanup	MIN_TRAJECTORY_LENGTH	direct pass-through	pre-merge fragment removal
Velocity break	MAX_VELOCITY_BREAK	body-lengths per second → pixels per frame	split on implausible motion
Occlusion split	MAX_OCCLUSION_GAP	direct pass-through	split and relink max-gap logic

UI area / control	Runtime keys	Transformation	Primary consumers
Tracklet relinking	ENABLE_TRACKLET_RELINKING, RELINK_POSE_MAX_DISTANCE	direct pass-through	motion-and-pose fragment reconnection
Velocity z-score logic	MAX_VELOCITY_ZSCORE, VELOCITY_ZSCORE_WINDOW, VELOCITY_ZSCORE_MIN_VELOCITY	minimum velocity scaled by s_{ref}/FPS	adaptive split detection
Forward/backward merge	AGREEMENT_DISTANCE, MIN_OVERLAP_FRAMES	agreement multiplier $\times s_{\text{ref}}$ plus integer overlap threshold	conservative consensus merge
Interpolation	INTERPOLATION_METHOD, INTERPOLATION_MAX_GAP	direct pass-through	short-gap fill after identity decisions
Visualization-free and stats	VISUALIZATION_FREE_MODE, ENABLE_HISTOGRAMS, HISTOGRAM_HISTORY_FRAMES, TRAJECTORY_HISTORY_SECONDS	direct pass-through	throughput mode, telemetry, and overlay history
Overlay toggles	SHOW_FG, SHOW_BG, SHOW_CIRCLES, SHOW_ORIENTATION, SHOW_YOLO_OBB, SHOW_TRAJECTORIES, SHOW_LABELS, SHOW_STATE, SHOW_KALMAN_UNCERTAINTY	direct pass-through	live preview and rendered-video overlays

Table 6: Pose, identity, and downstream dataset controls.

UI area / control	Runtime keys	Transformation	Primary consumers
Dataset generation toggles	ENABLE_DATASET_GENERATION, DATASET_NAME, DATASET_CLASS_NAME, DATASET_OUTPUT_DIR, DATASET_MAX_FRAMES, DATASET_CONF_THRESHOLD	direct pass-through	frame export for active-learning datasets

UI area / control	Runtime keys	Transformation	Primary consumers
Dataset sampling metrics	DATASET_DIVERSITY_WINDOW, DATASET_INCLUDE_CONTEXT, DATASET_PROBABILISTIC_SAMPLING, METRIC_LOW_CONFIDENCE, METRIC_COUNT_MISMATCH, METRIC_HIGH_ASSIGNMENT_COST, METRIC_TRACK_LOSS, METRIC_HIGH_UNCERTAINTY	direct pass-through	dataset export scoring and frame selection
Identity / individual analysis	ENABLE_IDENTITY_ANALYSIS, ENABLE_INDIVIDUAL_PIPELINE, IDENTITY_METHOD, IDENTITY_CROP_SIZE_MULTIPLIER, IDENTITY_CROP_MIN_SIZE, IDENTITY_CROP_MAX_SIZE	direct pass-through	crop extraction and downstream identity workflows
Tag-based identity options	COLOR_TAG_MODEL_PATH, COLOR_TAG_CONFIDENCE, APRILTAG_FAMILY, APRILTAG_DECIMATE	direct pass-through	tag-based identity extensions
Pose extractor enablement	ENABLE_POSE_EXTRACTOR, POSE_MODEL_TYPE, POSE_MODEL_DIR, POSE_RUNTIME_FLAVOR, POSE_EXPORTED_MODEL_PATH	model path resolution + runtime derivation	pose backend creation and precompute
Pose geometry definition	POSE_MIN_KPT_CONF_VALID, POSE_SKELETON_FILE, POSE_IGNORE_KEYPOINTS, POSE_DIRECTION_ANTERIOR_KEYPOINTS, POSE_DIRECTION_POSTERIOR_KEYPOINTS	direct pass-through from project controls	heading derivation and normalized pose prototype generation
Pose backend batching and env	POSE_YOLO_BATCH, POSE_BATCH_SIZE, POSE_SLEAP_ENV, POSE_SLEAP_DEVICE, POSE_SLEAP_BATCH, POSE_SLEAP_MAX_INSTANCES, POSE_SLEAP_EXPERIMENTAL_FEATURES	direct or runtime-derived	pose inference service and export mode selection

UI area / control	Runtime keys	Transformation	Primary consumers
Pose cache handoff	INDIVIDUAL_PROPERTIES_CACHE_PATH	path passed directly from UI state	pose direction override and association cues during tracking
Real-time crop export	ENABLE_INDIVIDUAL_DATASET, ENABLE_INDIVIDUAL_IMAGE_SAVE, INDIVIDUAL_DATASET_NAME, INDIVIDUAL_DATASET_OUTPUT_DIR, INDIVIDUAL_OUTPUT_FORMAT, INDIVIDUAL_SAVE_INTERVAL, INDIVIDUAL_INTERPOLATE_OCCLUSIONS, INDIVIDUAL_CROP_PADDING, INDIVIDUAL_BACKGROUND_COLOR	direct pass-through	crop extraction for PoseKit, ClassKit, or manual review

12.3 How operator controls move the math

The crosswalk tables above are comprehensive, but operators usually need a more causal reading: if a control moves, which mathematical object moves with it, and what does the tracker do differently?

Table 7: High-value UI controls and their mathematical effect.

UI control family	Mathematical target	Runtime effect
Reference body size and resize factor	s_{ref} and a_{ref}	They set the length and area scales used to derive MAX_DISTANCE_THRESHOLD, CONTINUITY_THRESHOLD, MIN_RESPAWN_DISTANCE, AGREEMENT_DISTANCE, MIN_OBJECT_SIZE, and MAX_OBJECT_SIZE. Doubling either control doubles most distance-like gates and quadruples area-like gates.
Assignment distance multiplier	d_{cull} and the accepted positional radius	This widens or tightens the candidate set before advanced costing. Large values reduce fragmentation but increase the chance of cross-animal confusion in dense scenes.
Recovery search multiplier	established-track split threshold	This changes which tracks are treated as established in the first assignment phase. Raising it makes the solver more conservative about giving tracks privileged first access to detections.
Kalman noise covariance	$\mathbf{Q}_t$ magnitude	Increasing it tells the filter to trust motion memory less and tolerate more maneuvering. That broadens the prediction cloud and makes association less brittle, but can also admit more false matches.

UI control family	Mathematical target	Runtime effect
Kalman measurement noise covariance	**R** magnitude	Increasing it tells the filter to trust detections less. The state then moves more slowly toward each measurement, which can smooth noisy detections but can lag behind abrupt real motion.
Kalman damping and initial velocity retention	**F** and young-track velocity state	These determine how far a prediction coasts when recent detections are missing. Strong damping and low early retention reduce overshoot but make tracks easier to fragment during brief dropouts.
Longitudinal and lateral noise multipliers	$\mathbf{Q}_v(\theta)$ anisotropy	These reshape the predicted uncertainty ellipse around the current body heading. Raising the longitudinal multiplier makes forward motion easier to absorb; raising the lateral multiplier makes sideways reassignments easier to accept.
Association weights	C_{ij} mixture weights	These decide whether motion, orientation, area, or aspect dominates the online score. For example, increasing `W\_ORIENTATION` makes the tracker care more about heading consistency during crossings.
Pose rejection threshold and visibility floor	pose veto inequality	Lower thresholds make the pose-shape veto more aggressive; higher visibility floors require cleaner keypoint evidence before the veto is allowed to reject a candidate.
Respawn multiplier	respawn exclusion radius	This sets how far a free detection must be from every non-lost prediction before a lost slot may be reused. Raising it suppresses duplicate track creation around crowded animals.
Merge agreement multiplier and overlap frames	forward/backward consensus test	These determine how much spatial and temporal agreement forward and backward fragments need before being merged. Conservative values reduce false merges at the cost of more residual fragments.
Velocity break and z-score controls	fragment split tests	These control when a track is judged dynamically implausible and cut into smaller fragments. They should be loosened only after detection and association are already credible.
Relink pose distance and interpolation gap	offline repair acceptance	These only act after primary identity decisions. Tight relink thresholds preserve conservative identity boundaries; larger interpolation gaps improve visual continuity but can hide uncertainty if used too early.

13 Operational Guidance

For non-specialist operators, the most important practical sequence remains:

1. validate detections before touching identity parameters,
2. set a realistic reference body size,

3. tune assignment and lifecycle thresholds next,
4. use backward replay and post-processing to resolve residual ambiguity,
5. and only then treat relinking and interpolation as refinement tools.

Three common failure classes map cleanly onto three intervention orders:

- **Frequent merges into one animal.** Improve detection and morphology first; do not just widen assignment gates.
- **Excessive fragmentation.** Relax loss tolerance and candidate gating modestly; verify that pose rejection is not too strict.
- **Identity swaps at crossings.** Enable backward replay, keep merge thresholds conservative, and use pose-derived direction only when the keypoints are genuinely reliable; otherwise prefer explicit marker identity if available.

14 Implementation Surface

The following code paths are the highest-value audit points for the current behavior:

Table 8: Primary implementation modules.

Module	Primary responsibility
`src/multi_tracker/core/tracking/worker.py`	run orchestration, detector control, detection-cache usage, frame iteration, lifecycle state updates, overlays, and signal emission
`src/multi_tracker/core/filters/kalman.py`	vectorized Kalman prediction / correction with anisotropic process noise, maturity damping, and Joseph-form updates
`src/multi_tracker/core/assigners/hungarian.py`	cost matrix construction, advanced gating, pose rejection, staged assignment, and lost-slot respawn logic
`src/multi_tracker/core/post/processing.py`	fragment cleaning, conservative forward/backward resolution, relinking, and interpolation
`src/multi_tracker/core/identity/runtime_api.py`	pose runtime construction shared across MAT and PoseKit
`src/multi_tracker/gui/main_window.py`	UI control collection, scaling from user multipliers to runtime values, and config compatibility logic
`src/multi_tracker/posekit/ui/main_window.py`	project-driven pose annotation, prediction, smart selection, and evaluation flows
`src/multi_tracker/classkit/gui/mainwindow.py`	active-learning classification workbench built on embeddings and persistent project state
`src/multi_tracker/tools/data_sieve/core.py`	deduplication, temporal subsampling, and diversity-oriented dataset filtering

Module	Primary responsibility
`tests/test_tracking_pipeline_synthetic.py`	synthetic regression for motion, assignment, and post-run resolve/interpolate behavior
`tests/test_post_tracklet_relinking.py`	targeted regression checks for motion-and-pose relinking

15 Summary

The current MAT tracker is best described as a conservative, modular tracking system rather than a single closed-form tracker. It deliberately combines:

- online motion prediction,
- geometry-aware and pose-aware association,
- lifecycle state transitions,
- replay-based hypothesis comparison,
- and offline continuity repair.

That architecture is more operationally complex than a minimal tracker, but it is also why the present codebase can bridge from raw video to tracking, pose labeling, classification dataset construction, and dataset curation without treating those downstream tasks as an afterthought.

The next major gains are likely to come from stronger explicit identity channels, better offline fusion, and faster staged detection, not from simply increasing the weight of today's pose or appearance cues inside the online Hungarian assignment.