ANSODEngine — Tracker & Stabilization Developer Guide

Module: **ANSODEngine** · Base class: **ANSCENTER::ANSODBase** · Affects every OD engine subclass (ANSRTYOLO, ANSONNXYOLO, ANSYOLOV12RTOD, etc.)

1. Default state after model load

Every engine derived from **ANSODBase** finishes `Initialize` / `LoadModel` / `LoadModelFromFolder` with these defaults:

Feature	Default	Source of default
Tracker (BYTETrack / UCMC / OCSORT)	**ON**	Loaders auto-call SetTracker(BYTETRACK, true) (ANSODEngine.cpp:77, 110, 139, 174, 246, 260)
Stabilization (ghost injection + EMA + hysteresis + class consistency)	**OFF**	Member initializer _stabilizationEnabled = false; no auto-enable

Notes. Tracker assigns a `trackId` to detections and (when stabilization is on) lets the tracker predict bridging "ghost" boxes. Stabilization is opt-in and requires the tracker. It cannot run standalone.

2. Public setters — invariant table

The single invariant maintained by the API: **stabilization requires tracker** (`stab=true ⇒ tracker=true`).

Call	_trackerEnabled	_stabilizationEnabled	Why
`SetTracker(type, true)`	→ true	unchanged	Tracker is independently useful. Don't side-effect stab.
`SetTracker(type, false)`	→ false	→ **false** (forced)	Stabilization can't run without tracker. Forced clear keeps the flag honest.
`SetStabilization(true, hist, miss)`	→ **true** (auto if off)	→ true	Stabilization needs tracker. Auto-enables it.
`SetStabilization(false)`	unchanged	→ false	Tracker stays usable on its own.
`SetStabilizationParameters(json)`	unchanged	unchanged	Tunes thresholds only; does not toggle either flag.
`SetTrackerParameters(json)`	unchanged	unchanged	Tunes tracker params only; no-op if tracker is off.

3. Reachable runtime states

_trackerEnabled	_stabilizationEnabled	What runs at inference
`false`	`false`	Raw detections only. No trackId. No ghosts.
`true`	`false`	Detections + trackId assignment. **No ghosts, no EMA, no hysteresis.** ← default
`true`	`true`	Full pipeline: tracking + ghost injection + EMA confidence + hysteresis + class consistency + stale-history pruning.
`false`	`true`	**Unreachable** — invariant prevents it.

4. How to opt in / out (caller recipes)

Goal	Sequence
Default (tracker only, no stabilization)	`Nothing. Just load the model.`
Tracker only, explicit	`SetTracker(BYTETRACK, true)`
Tracker + stabilization	`SetStabilization(true, 20, 5)  (auto-enables tracker)`
Tracker + stabilization with custom thresholds	`SetStabilization(true) then SetStabilizationParameters(json)`
Disable everything	`SetTracker(BYTETRACK, false)  (also clears stabilization)`
Switch tracker type without losing stabilization	`SetTracker(UCMC, true) then SetStabilization(true) if it was on`

5. Stabilization parameters

Apply only when stabilization is on. `SetStabilization(enabled, historySize, maxMisses)` arguments:

Param	Default	Min / clamp	Effect
`historySize`	20	≥ 5	Frames retained in _detectionQueue per camera.
`maxMisses`	5	≥ 1	Frames a ghost may persist before pruning. History pruned at 2×maxMisses unseen frames.

`SetStabilizationParameters(json)` — all keys optional:

Key	Default	Range	Purpose
`hysteresis_enter`	model threshold	float, 0 = auto	Confidence needed for a *new* track to appear.
`hysteresis_keep`	0.65 × enter	float, 0 = auto	Lower threshold to *keep* an established track visible.
`ema_alpha`	0.30	[0.01, 1.0]	EMA weight for new observations (lower = smoother).
`track_boost_min_frames`	10	≥ 1	Frames before a track is “established” and gets confidence boost.
`track_boost_amount`	0.05	[0.0, 0.5]	Confidence bonus for established tracks.
`class_consistency_frames`	5	≥ 1	Consecutive frames of a new class required before accepting a class switch on an established track.
`confidence_decay`	0.85	[0.1, 1.0]	Per-miss multiplier on ghost confidence.
`min_confidence`	0.15	[0.0, 1.0]	Floor below which ghosts are dropped.

6. Performance & memory characteristics

Aspect	Tracker only	Tracker + Stabilization
Per-frame CPU cost	BYTETrack association, ~O(N·M)	+ EMA / hysteresis / class-consistency loop O(N) + history-prune O(K)
Mutex acquisitions per inference	1 extra inside ApplyTracking	2 extra (ApplyTracking + StabilizeDetections) on the engine’s _mutex
Result vector size	= real detections	= real detections + ghost objects in grace window
Per-camera state	ANSMOT instance + BYTETrack removed_stracks_ (capped at 1000)	+ _trackHistories (pruned at unseen > 2×maxMisses) + _detectionQueue (capped at historySize)
Bounded?	Yes — caps in BYTETrack (kRemovedCap=1000)	Yes — explicit prune in StabilizeDetections Phase 3

7. Diagnostic logging

Every flag transition emits a DebugView line via `ANS_DBG("ANSOD_StabState", …)`:

```
[ANSOD_StabState] this=<engine_ptr> SetTracker(true,type=N) tracker=0->1 stab=0 (unchanged)
[ANSOD_StabState] this=<engine_ptr> SetTracker(false) tracker=1->0 stab=1->0
[ANSOD_StabState] this=<engine_ptr> SetStabilization(true,hist=20,miss=5) tracker=1->1 stab
```

```
=0->1
[ANSOD_StabState] this=<engine_ptr> SetStabilization(false) tracker=1 (unchanged) stab=1->0
```

Capture with DebugView (or any ANSCORE log) and `grep ANSOD_StabState` to audit which engine instances run with which flags in production.

8. Behavior change vs. previous build

For developers updating from the prior version of ANSCORE.

Behavior	Before	After
SetTracker(true) side-effect on stabilization	Auto-set stabilization to true	**No effect on stabilization**
SetTracker(false) side-effect on stabilization	None (could leave stab=true while tracker=false)	**Forced stabilization to false**
Default after model load	tracker=ON, stabilization=ON	tracker=ON, **stabilization=OFF**
SetStabilization(true) auto-enable tracker	Yes	Yes (unchanged)
SetStabilization(false) side-effect on tracker	None	None (unchanged)

Migration note. If your application relied on the old auto-enable to get stabilization for free, add an explicit `SetStabilization(true)` call after model load. ANSLPR's defensive `SetStabilization(false)` calls remain correct (now redundant but harmless).

This guide reflects the ANSODEngine API as of the 'tracker/stabilization invariant' cleanup. File references point to modules/ANSODEngine/ANSODEngine.cpp and modules/ANSODEngine/ANSEngineCommon.h.