Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

CLASP on CPUs: Measured Energy, Deadline Compliance and Exact Computation Skipping for Multi-Person Tracking

SANTIAGO BAQUERO DELGADO

sbaquero66@unisalle.edu.co

Research Article

Keywords:

Posted Date: November 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11495182/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

CLASP on CPUs: Measured Energy, Deadline Compliance and Exact Computation Skipping for Multi-Person Tracking

Santiago Baquero Delgado

Received: date / Accepted: date

Abstract Multi-person tracking on CPU-only platforms is usually reported in frames per second, with energy inferred rather than measured. We measure per-stage latency, per-frame deadline compliance, cores used and Intel RAPL package energy of five tracking-by-detection methods—CLASP, ByteTrack, DeepSORT, the official OC-SORT and TrackTrack—with isolated processes, seeded run orders, idle baselines and hash-verified inputs and outputs, over 76 sequences of MOT17, MOT20 and DanceTrack and three detector scales. We also study *exact* computational reduction of association: a component-wise dominance certificate decides in linear time when a bounded appearance term cannot change the optimal assignment, and four exact reformulations remove redundant work. The certified CLASP produces byte-identical output on 158 sequences at 0.64× the tracker energy of its reference. We prove that the certificate can never succeed once the appearance weight reaches the acceptance threshold, and measure the consequence: revising CLASP for accuracy raised its tracker energy by 37% (95% CI 22–53%), turning a tracker as cheap as ByteTrack into one 1.4× as expensive. Association uses one core and meets a 30 fps deadline with a margin above 16×. With YOLO11n in the loop, lightweight trackers average 30 fps, yet 36–46% of frames exceed 33.3 ms; camera-motion compensation raises system energy by 43% and removes that operating point. A vectorised cost matrix reduced operations but increased energy. Scheduling the detector over time cost accuracy, and crop verification saved 21–41% on design sequences but failed its pre-registered held-out confirmation.

S. Baquero Delgado
Universidad de La Salle, Bogotá, Colombia
E-mail: sbaquero66@unisalle.edu.co

Keywords Multi-object tracking · Energy measurement · CPU-only inference · Deadline compliance · Intel RAPL · Linear assignment

1 Introduction

Tracking-by-detection links per-frame detections into identities [3,29,31,5]. The computational footprint of the detectors is well studied; that of the *tracking* stage is almost never measured, and tracker efficiency is reported as frames per second on a GPU workstation [31, 1]. Three facts make this inadequate for real-time deployments. First, many systems—traffic and retail counting, ambient assisted living, low-cost robots—run CPU-only, where every millisecond is energy. Second, as detectors shrink for the edge, the relative weight of the tracker grows, and trackers that add camera-motion compensation or appearance re-identification can consume a large share of the frame budget. Third, the Green AI agenda [22,25,10,20] asks for energy to be *measured*; operation counts are an unreliable proxy on modern CPUs, where memory traffic, interpreter overhead and vectorisation dominate.

Methods for cheaper tracking are either *lossy* (skip frames, lower resolution, drop features) or *architectural* (redesign the tracker, so accuracy must be re-established). Neither guarantees an unchanged output. This paper combines a measurement study with a third route, *exact computational reduction*: we prove, per frame, that an expensive part of the association cannot change the result, and skip it—and we characterise when such proofs are possible.

Research questions. (RQ1) What do CPU-only trackers cost per frame in latency, deadline compliance and

package energy, measured rather than estimated, and how does this cost relate to accuracy? (RQ2) Can the association stage be made cheaper with no change in output, when, and does the saving appear in measured energy? (RQ3) Where does the energy of a complete CPU pipeline go, and how does the tracker's share change with detector scale? (RQ4) Can the detector, which dominates the energy, be scheduled in time or in space without a demonstrable loss of identity accuracy?

Contributions. (i) A reproducible CPU-only protocol: fresh process per run, core pinning, seeded orders, idle baselines, RAPL windows, hash-verified identity of inputs and reproducibility of outputs, and a run manifest of every accepted run. (ii) A paired bench that resolves $\sim$5% differences in sub-millisecond tracker energy. (iii) A dominance certificate for bounded-perturbation assignment with proof, a necessary condition that predicts its yield from the appearance weight, and an exact implementation of CLASP verified byte-for-byte on 158 sequences. (iv) A system-level decomposition over three detector scales and a deadline analysis under two explicit workloads, including per-frame deadline-miss rates and the number of cores actually used. (v) A paired measurement of the energy regression introduced when CLASP was revised for accuracy, with its cause. (vi) A live, energy-measured study of temporal and spatial detector scheduling, including a failed pre-registered confirmation, and other measured negative results.

2 Related work

Trackers. SORT [3] introduced Kalman prediction [13] with Hungarian assignment [15]; DeepSORT [29] added a deep appearance metric; ByteTrack [31] associates low-score detections in a second stage; OC-SORT [5] re-updates the filter from observations after occlusion; BoT-SORT [1] and TrackTrack [24] add global motion compensation (GMC); StrongSORT [8] and Deep OC-SORT [17] add stronger ReID. Joint detection-and-embedding models [27,32] amortise appearance inside the detector. Transformer trackers [30,33] need a GPU and are out of scope.

Energy measurement. Intel RAPL exposes model-specific energy counters for the processor package, cores and, where available, DRAM [7]; its accuracy and pitfalls—counter granularity, update period, domain coverage—have been characterised [9,14]. Reporting guidelines for machine-learning energy [10] ask for physical measurement, repetition and explicit baselines; we follow them and add paired designs against slow drift of background power.

Assignment sensitivity. The linear assignment problem and its sensitivity analysis are classical [4]; robust assignment under interval costs is NP-hard in general. Our certificate is not a robust solver: it is a sufficient, linear-time test of whether an uncertain term can change the optimum, used to decide whether to compute it.

3 Trackers and the certified lazy association

3.1 CLASP in brief

CLASP (Confidence-Aware Lightweight Association and Spatial Persistence) is a learning-free tracker whose design and accuracy are the subject of a companion paper [2]. Per frame it solves a cascade of linear assignment problems between tracks and detections. For track i (last box b_i, height h_i) and detection j inside a size-adaptive gate of radius r_i, the geometric cost is

$$g_{ij} = w_d \frac{\|\mathbf{c}_j - \mathbf{c}_i\|}{r_i} + w_o\big(1 - \mathrm{IoU}(b_i, b_j)\big) + w_s \frac{\min(|\ln(A_j/A_i)|, 1.5)}{1.5} + w_\theta \frac{1 - \cos\theta_{ij}}{2}, \tag{1}$$

with weights $(0.5, 0.3, 0.1, 0.1)$. A pair is *ambiguous* if its row or column has more than one in-gate entry; only ambiguous pairs receive an appearance term $a_{ij} = w_a\, d_B(H_i, H_j) \in [0, w_a]$, the Bhattacharyya distance [12] between 16×16 hue–saturation histograms [26]. Entries with $c_{ij} = g_{ij} + a_{ij} > c_{\max} = 0.65$ are masked with a large constant M and a minimum-cost assignment [11,6] is solved; masked pairs are discarded. The final CLASP uses $w_a = 0.7$, an observation-centric track history and track-aware initialisation; its first, frozen version used $w_a = 0.3$ and neither rule [2]. We report the final version and use the frozen one where the appearance weight matters. Profiling the reference implementation (CLASP-Ref) shows most of the update time in histogram computation and in recomputing a per-track velocity once per *pair*.

3.2 Interval bounds

Before any histogram is computed, each candidate entry is bounded: $l_{ij} = g_{ij}$ and $u_{ij} = g_{ij} + w_a$ if the pair is ambiguous, $u_{ij} = g_{ij}$ otherwise. Ambiguity depends only on the gate, so the bounds are exact. The candidate set is $K = \{(i,j) : l_{ij} \le c_{\max}\}$; a pair outside K is masked for every value of $a_{ij} \ge 0$, so its histogram is never needed (*candidate filtering*).

3.3 Dominance certificate

Proposition 1 *Let M exceed the sum of all admissible costs. Suppose that for every row i with candidates there is a column $j^*(i)$ such that (a) $u_{ij^*(i)} \le c_{\max}$; (b) $u_{ij^*(i)} < l_{ij}$ for every other candidate j of row i; and (c) the columns $j^*(i)$ are pairwise distinct. Then for every realisation $a_{ij} \in [0, w_a]$ of the ambiguous entries, the set of admissible pairs returned by an optimal solution of the masked assignment problem is exactly $\{(i, j^*(i))\}$.*

Proof Fix a realisation. Since M exceeds the sum of admissible costs, an optimal assignment first maximises the number of admissible pairs and then minimises their cost. By (a), each $(i, j^*(i))$ is admissible; by (b), for any other candidate j of row i, $c_{ij} \ge l_{ij} > u_{ij^*} \ge c_{ij^*}$, so $j^*(i)$ is the strict minimum of its row among admissible entries. By (c), $\{(i, j^*(i))\}$ is feasible and matches every row with candidates, attaining the maximum number of admissible pairs. Among assignments attaining that maximum, each such row is matched admissibly and its cost is at least its row minimum, with equality only at $j^*(i)$; the sum is therefore uniquely minimised by $\{(i, j^*(i))\}$. □

In the implementation $M = 10^6$, while every admissible cost is at most $c_{\max}$ and a stage never has more than a few dozen admissible pairs, so the hypothesis on M holds with a margin of four orders of magnitude. Admissible pairs form a bipartite graph whose minimum-cost maximum-cardinality matching decomposes over connected components, so the certificate is checked per component in $O(|K|)$ and histograms are computed only in components that fail it. One histogram costs 40–100 µs on our CPU. Tie-breaking inside the solver for non-certified components and floating-point effects are not covered by the proof, which is why equivalence is also verified empirically.

When can the certificate succeed? In a connected component with at least two candidate pairs every pair shares a row or a column with another, so every pair is ambiguous. Condition (a) then requires $g_{ij^*} + w_a \le c_{\max}$ for the chosen entry of every row:

Corollary 1 *A connected component with more than one candidate pair can be certified only if each of its rows has a candidate with $g_{ij} \le c_{\max} - w_a$. In particular, if $w_a \ge c_{\max}$ no such component is ever certified, and the fraction of certifiable components is non-increasing in w_a.*

The yield of the certificate is therefore a property of the operating point, not only of the data: it measures how often the expensive cue *cannot* change the decision. We report it as a function of w_a in Sect. 5.2.

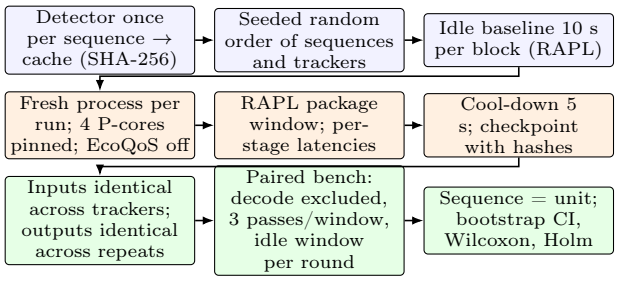


Fig. 1 Measurement protocol. Top: preparation; middle: one measured run; bottom: verification, tracker-level bench and inference

3.4 Exact reformulations

E1 memoises the per-track velocity, which the reference recomputes for every pair. **E2** memoises each histogram per (frame, box), which the reference recomputes across cascade stages and updates. **E3** is the certificate with candidate filtering. **E4** discards a pair when $|d_x| > r$ or $|d_y| > r$ before evaluating the Euclidean norm, since $\|\mathbf{d}\|_2 \ge \|\mathbf{d}\|_\infty$. **T** replaces the $O(K)$ mean of the last K centre differences by its telescoped form $(\mathbf{c}_n - \mathbf{c}_{n-K})/K$, exact in real arithmetic; its floating-point effect is verified, not assumed, and it subsumes E1. The certified CLASP is E2+E3+E4+T. Two *approximate* variants (lazy appearance refresh, 2× sub-sampled histograms) were evaluated against a pre-registered non-inferiority margin and are not part of the method.

3.5 Comparison trackers

ByteTrack and TrackTrack (the latter with sparse optical-flow GMC [23]; neither with ReID) use the Ultralytics 8.4.115 implementations; OC-SORT uses the authors' code [5]; DeepSORT uses deep-sort-realtime 1.3.2 with a MobileNetV2 embedder [21]. The thresholds of all four were calibrated with eight configurations each on the eight design sequences of CLASP and frozen [2].

4 Real-time and energy evaluation protocol

Figure 1 summarises the protocol.

4.1 Platform and isolation

All runs use an Intel Core i9-14900HX laptop CPU (hybrid: 8 performance and 16 efficiency cores) with 32 GB RAM, Windows 11 (build 26200, "Balanced" power scheme, mains power), Python 3.12.10, a CPU-only build of PyTorch 2.13.0 (no CUDA), OpenCV 5.0.0,

NumPy 2.4.6, SciPy 1.18.0 and Ultralytics 8.4.115. Every measured process is pinned to four logical CPUs on distinct performance cores with Windows power throttling (EcoQoS) disabled, and runs four threads for PyTorch, OpenCV and BLAS. Temperature was not logged; thermal stability is assessed indirectly through the repeat-to-repeat coefficient of variation of throughput and the per-block idle power.

4.2 Workloads and deadlines

We distinguish two workloads, both with the RAPL package boundary of Sect. 4 and two repeats per run. **W1, association only**: detections are available from a cache (equivalently, from an accelerator or a shared detector); the measured loop decodes the frame, reads its detections and runs the tracker. **W2, complete pipeline**: the detector runs in the loop on the CPU. Latency is split into decoding, detection or cache access, and tracking, and reported as median, 95th and 99th percentile per run. We distinguish *mean capacity* (frames per second over a run) from *deadline compliance*. A tracker meets a deadline D in W1 when its 99th-percentile latency is below D (conservative criterion). In W2 we report the fraction of frames whose latency exceeds D, from per-frame latencies recorded in a dedicated bench (Sect. 5.7). Deadlines are 33.3 ms (30 fps) and 40 ms (25 fps). Four threads are *allowed* to PyTorch, OpenCV and BLAS; the trackers themselves are single-threaded Python, and we report the effective number of cores used (process CPU time divided by wall time).

4.3 Energy boundaries

Energy is read from the Intel RAPL package counter [7] at the start and end of every measured window (Windows Energy Metering Interface). This is the *only* energy boundary used: no external wall-power meter was available, and we do not report wall energy. The package domain covers cores, caches and the integrated GPU plane; the DRAM domain is not exposed on this platform. We report gross energy and energy net of the idle power measured for 10 s before each block. A model-based estimate (TDP × utilisation) is logged by the pipeline but is *not* energy and is not reported.

4.4 Tracker-level paired bench

For sub-millisecond workloads a whole-run window is dominated by decoding. We therefore decode the frames of eight sequences (300 frames each; 2 283 frames) into memory once, and within one RAPL window each tracker processes the clip three times with a fresh instance. In every round the trackers and an idle window of equal scale run in a seeded random order; the idle power of that round is subtracted to obtain dynamic energy. Seven rounds per sequence; ratios to CLASP-Ref are formed within (sequence, round) and averaged geometrically; the sequence is the unit of inference (95% bootstrap CI, 5 000 resamples; Wilcoxon signed-rank test [28]).

4.5 Pipeline runs

Each (sequence, tracker, repeat) runs in a fresh process. Sequence order and tracker order within each block are seeded random, a 10 s idle baseline precedes each block, and a 5 s cool-down follows each run. Inputs are verified identical across trackers through SHA-256 digests of detections and frames; outputs are required to be byte-identical across repeats. Every run is checkpointed with hashes of its outputs and telemetry, including the power source at the start and end of the window; a run is accepted only if the platform was on mains power throughout, and a guard prevents a run from starting otherwise (130 runs of the original campaign that executed on battery after a charger disconnection were discarded and repeated). CLASP, ByteTrack, OC-SORT and TrackTrack were measured in the same blocks; DeepSORT, two orders of magnitude more expensive, was measured in the original campaign under the same protocol. Accuracy is computed with TrackEval [16] and taken from the companion paper; here it serves only to place costs in context.

4.6 Equivalence

Every optimised variant runs frame by frame next to CLASP-Ref on identical inputs, and the emitted (identity, box) lists are compared at every frame; final MOT files are compared byte-for-byte.

5 Results

5.1 Output equivalence (RQ2)

The certified CLASP (E2+E3+E4+T) produced output identical to CLASP-Ref at every frame on 158 of 158 sequences: the 76 ground-truth and 46 test sequences with YOLO11m detections and 36 sequences with public FRCNN and SDP detections (166 380

frames). Every intermediate configuration of the cumulative bench was also identical on the bench sequences. Accuracy is therefore unchanged by construction. The frozen version had been verified in the same way on 140 sequences [2].

5.2 Certificate yield (RQ2)

Figure 2 reports, on the bench sequences, the fraction of connected components with ambiguous candidate pairs that the certificate decides without any histogram, and the histograms computed by the certified implementation relative to the reference, as a function of the appearance weight (output identical at every weight). At the frozen operating point ($w_a = 0.3$) the certificate decides 79% of the components and the histograms fall to 0.45 of the reference; at $w_a = 0.5$, 61%; at $w_a = 0.6$, 25%; and, as the corollary predicts, essentially none from $w_a = c_{\max} = 0.65$ on, including the final $w_a = 0.7$, where the remaining saving (0.75 of the histograms, −37.5% appearance comparisons over the 158 sequences) comes from candidate filtering and memoisation. A refinement that computes the histograms of the row-best columns first and re-checks the certificate decided 95% of the components at $w_a = 0.7$ but saved only 0.7% of the histograms, because the row-best columns cover almost all ambiguous detections; it is not part of the method.

The two accuracy results of the companion paper explain this curve. With $w_a = 0.3$ the appearance cue could change the assignment in only a fifth of the ambiguous components, and removing it did not change HOTA significantly; with $w_a = 0.7$ it can change every ambiguous decision, and removing it raises false positives by 16%. *The certificate measures, exactly, how often an expensive cue is decision-relevant*: it is free when the cue is weak and has nothing to skip when the cue matters.

5.3 Tracker-level energy (RQ2)

Figure 3 shows the cumulative effect of the exact mechanisms on the final CLASP (7 rounds × 3 passes; every step identical to the reference). E1 and E2 together reach 0.75× [0.72, 0.79]. Adding the certificate raises energy to 0.86×: at $w_a = 0.7$ it never succeeds and its check is pure overhead—a measured confirmation of the corollary. The L_∞ pre-filter and the telescoped velocity bring the total to 0.66×, and the certified CLASP (E1 dropped as subsumed by T) reaches 0.64× [0.56, 0.73] of the reference energy (8/8 sequences, $p = 0.008$). At the frozen operating point the same layer reached 0.49×, the certificate being its largest single contribution. Table 1 compares the trackers in the same rounds: the certified CLASP (1.00 ms, 26.6 mJ per frame) costs 1.4× ByteTrack and OC-SORT and 23× less than TrackTrack, whose cost is dominated by optical-flow GMC on every frame. Skipping the certificate check when $w_a \geq c_{\max}$, which the corollary justifies a priori, would remove the overhead measured at step E3.

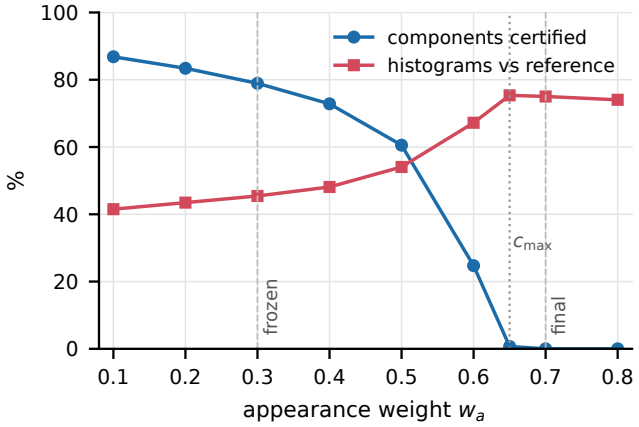


Fig. 2 Yield of the dominance certificate versus appearance weight w_a (8 bench sequences × 300 frames): components with ambiguous candidate pairs decided without histograms, and histograms computed relative to the reference. The output is identical at every weight. Dotted: $w_a = c_{\max}$

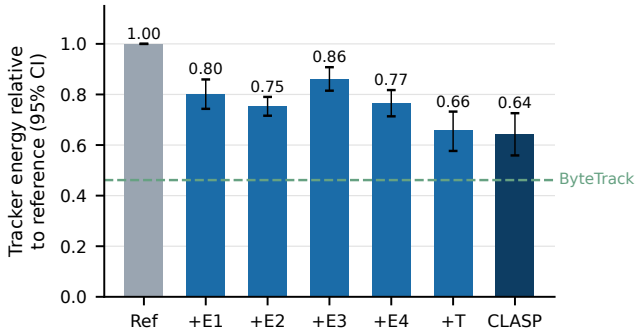


Fig. 3 Cumulative effect of the exact mechanisms on the energy of the final CLASP (paired RAPL bench, 8 sequences, 7 rounds; 95% bootstrap CI). Ref: CLASP-Ref; E1: velocity memo; E2: histogram memo; E3: certificate and candidate filtering; E4: L_∞ pre-filter; T: telescoped velocity; CLASP: E2+E3+E4+T. Every bar produces output identical to the reference

Table 1 Tracker-level paired bench (8 sequences × 300 frames, 7 rounds × 3 passes, decoding excluded). Ratios relative to CLASP-Ref with 95% bootstrap CI; Wilcoxon over sequences

Tracker	ms/fr.	mJ/fr.	Ratio [95% CI]	p (lower/8)
CLASP-Ref	1.43	39.4	1.00	–
CLASP (certified)	1.00	26.6	0.64 [0.56, 0.73]	0.008 (8/8)
ByteTrack	0.67	19.3	0.46 [0.38, 0.57]	0.008 (8/8)
OC-SORT	0.72	19.7	0.48 [0.43, 0.55]	0.008 (8/8)
TrackTrack	19.64	613.5	14.83 [12.19, 17.80]	0.008 (0/8)

Table 2 Energy regression between the versions of CLASP (paired RAPL bench, 8 sequences × 300 frames, 7 rounds × 3 passes, decoding excluded; ms and mJ per frame). cert.: certified (exact) implementation; Ref: reference code path; lite: CLASP-lite. Ratios relative to the frozen, certified CLASP (95% bootstrap CI); W: median package power; p: Wilcoxon over sequences (sequences with lower energy than the reference). ‡Not adopted

Variant	ms	mJ	W	Time	Energy [95% CI]	p (lower)
frozen, cert.	0.71	20.7	28.7	1	1	−−
frozen, Ref	1.42	41.1	28.7	2.00	1.98 [1.85, 2.10]	0.008 (0/8)
final, Ref	1.47	43.1	29.1	2.07	2.05 [1.91, 2.20]	0.008 (0/8)
final, cert.	1.03	28.7	27.5	1.43	1.37 [1.22, 1.53]	0.008 (0/8)
lite, Ref	1.43	42.5	29.3	2.02	2.04 [1.91, 2.17]	0.008 (0/8)
lite, cert.	1.01	29.0	28.6	1.40	1.37 [1.23, 1.51]	0.008 (0/8)
custody‡	1.20	33.9	28.0	1.68	1.61 [1.51, 1.71]	0.008 (0/8)
ByteTrack	0.69	20.2	28.4	0.94	0.92 [0.70, 1.21]	0.95 (5/8)
OC-SORT	0.74	21.8	29.4	1.04	1.05 [0.87, 1.26]	0.74 (3/8)

5.4 Did the revision of CLASP cost energy? (RQ2)

The final CLASP was adopted for its accuracy [2], and its larger appearance weight disables the certificate (Sect. 5.2). Table 2 measures the consequence in a single paired bench whose reference is the frozen, certified CLASP of the original study (same 8 sequences, 7 rounds × 3 passes). The final certified CLASP needs 1.37× [1.22, 1.53] the tracker energy of the frozen one and 1.43× its time (lower on 0 of 8 sequences, $p = 0.008$): *the revision is an energy regression of 37%.* Its cause is the exact-computation layer, not the association rules. Through the reference code path the two versions differ by 3% (2.05× versus 1.98× the frozen certified implementation), so the layer saved 50% on the frozen model and saves 33% on the final one. Mean package power is the same for every variant (27–29 W), so energy follows time. CLASP-lite costs as much as the final model (1.37×): the three heuristics it removes rarely execute. The frozen CLASP was as cheap as ByteTrack (0.92× [0.70, 1.21]) and OC-SORT (1.05× [0.87, 1.26]); the final one costs about 1.4× and 1.3× as much as they do. The reciprocal-custody rule piloted in the companion paper would add a further 18% (1.61×) and was rejected for accuracy. The trade-off is therefore explicit. On the held-out sequences the revision bought 1.4 HOTA for 37% more tracker energy, which is 0.5% of system energy even with YOLO11n (Table 4); on never-observed sequences it bought 1.7 HOTA, a difference that is not significant [2].

Negative result: fewer operations, more energy. A first implementation of the frozen CLASP vectorised the cost matrix with NumPy. It performed fewer interpreted operations yet was slower and more energy-hungry than the reference: +6% time and +9% energy ($p = 0.04$). With 5–25 detections per frame, array creation and dispatch overhead exceed the arithmetic saved. The scalar reformulation adopted instead hoists per-track quantities out of the pair loop. Operation counts are not a proxy for energy at this scale.

5.5 Association-only workload: latency, deadlines and energy (RQ1)

Table 3 (plotted in Supplementary Fig. S2) summarises the 760 whole-sequence runs of W1 (76 sequences, two repeats, decoding included): 608 runs of CLASP, ByteTrack, OC-SORT and TrackTrack measured in the same blocks of the final campaign and 152 runs of DeepSORT from the original campaign. The three lightweight trackers meet the 30 fps deadline with large margins at the 99th percentile (CLASP 2.0 ms, OC-SORT 1.5 ms, ByteTrack 1.2 ms); CLASP is the slowest of the three at the median (0.95 versus 0.64 and 0.77 ms). TrackTrack meets the deadline with a 1.8× margin (19 ms) and DeepSORT exceeds both deadlines (p99 219 ms). The worst single run of CLASP reaches 5.0 ms at the 99th percentile, still a 6.7× margin. Although four threads are allowed, the lightweight trackers use one core (process CPU 98–99%), TrackTrack 1.5 cores and DeepSORT 3.9; the margins above are therefore those of a single core. Net package energy of the whole W1 loop is 119.9, 120.5 and 124.4 mJ per frame for OC-SORT, ByteTrack and CLASP, with overlapping confidence intervals: decoding dominates the loop, and the three form one cost class, 4.1× below TrackTrack and 46× below DeepSORT. The 46 test sequences (runtime only) reproduce this picture.

5.6 Complete pipeline and detector scale (RQ3)

Table 4 reports W2 with the detector in the loop at three scales. With YOLO11m (MOT17 train) the median frame takes 134 ms with a lightweight tracker and no pipeline exceeds 7.2 fps. With YOLO11s throughput reaches 16.4 fps, and with YOLO11n (MOT17 and MOT20 train) 27.6–28.0 fps for the three lightweight trackers: the median frame takes 33.7–33.9 ms and the 95th percentile 37.5–37.7 ms, so a 25 fps (40 ms) deadline is met at the 95th but not at the 99th percentile (42.2–42.8 ms), and 30 fps is not met. TrackTrack, whose GMC adds about 19 ms per frame, stays at 18.5 fps with YOLO11n. At every scale the system energy of the lightweight trackers differs by at most 1.3% (7.20–7.27, 3.01–3.05 and 1.65–1.67 J per frame for YOLO11m, s and n), while TrackTrack raises system energy by 8%, 21% and 43% respectively; DeepSORT, measured in the original campaign, more than doubles

Table 3 Association-only workload (W1): 76 sequences, 2 repeats, decoding included. Latencies are medians over runs of the per-run statistic; CI = 95% bootstrap over sequences. HOTA: mean per sequence on the 68 held-out sequences, from the companion paper. DeepSORT measured in the original campaign

		Tracking latency (ms)			Deadline	Net energy (mJ/frame)		CPU	RAM	HOTA
Tracker	FPS	median	p95	p99	33.3 ms	mean	95% CI	(%)	(GB)	
CLASP	156.6	0.95	1.64	2.00	met	124.4	[113.7, 131.0]	98.5	0.37	39.51
ByteTrack	157.9	**0.64**	**0.82**	**1.19**	met	120.5	[110.5, 127.7]	97.9	0.36	33.96
OC-SORT	155.9	0.77	1.06	1.49	met	**119.9**	[110.0, 126.9]	97.9	0.39	40.94
TrackTrack	50.0	15.55	17.79	18.97	met	510.9	[479.5, 537.6]	152.5	0.37	40.34
DeepSORT	6.9	112.49	194.49	219.29	missed	5 700.8	[5 110.4, 6 401.4]	389.3	2.50	33.61

it with YOLO11m (15.6 J per frame). These end-to-end measurements confirm directly the trend of Table 5: the smaller the detector, the larger the share of the energy budget that a heavy tracker claims.

Table 5 (plotted in Supplementary Fig. S1) combines the measured detector cost at three scales with the tracker-level energies of Table 1. With a lightweight tracker the tracker's share of system energy stays at or below 1.4% at every scale; with GMC it grows from 5.9% (YOLO11m) to 24.3% (YOLO11n). The accuracy order of the strongest trackers changes with detector scale [2]: with YOLO11n TrackTrack is the most accurate (HOTA 34.1 on 36 sequences, against 32.9 for CLASP and 32.8 for OC-SORT), but it claims a quarter of the system energy and lowers throughput from 28 to 18.5 fps. At the edge, the choice of tracker is an accuracy–energy decision, and exact association savings are the only savings that come with no accuracy trade-off.

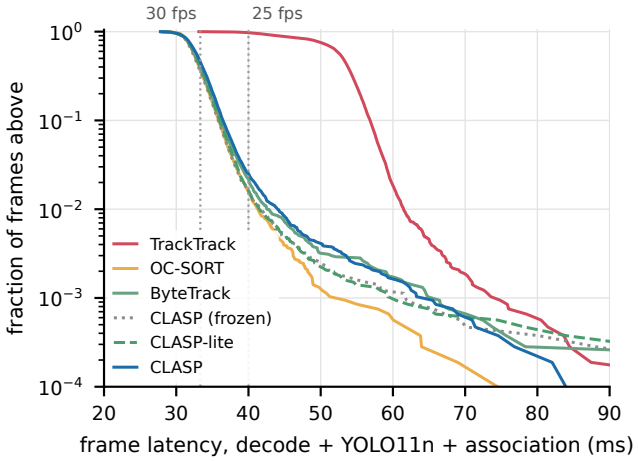


Fig. 4 Fraction of frames whose end-to-end latency (decoding, YOLO11n, association) exceeds a given value, MOT17 train, all frames of two repeats. Dotted lines: 25 and 30 fps deadlines

5.7 Per-frame deadline compliance of the complete pipeline (RQ1, RQ3)

Percentiles per run do not say how many frames miss a deadline. A dedicated bench therefore recorded every frame of the YOLO11n pipeline on MOT17 train. It covers the four trackers of Table 4 and two versions of CLASP: the frozen one and CLASP-lite, run on the certified path. Each tracker ran twice per sequence, in fresh pinned processes, in a seeded order within blocks preceded by idle baselines (Table 6, Fig. 4).

Mean capacity is not deadline compliance. Every lightweight tracker averages 29.8–30.2 fps, yet its median frame takes 32.6–33.1 ms. This is the 30 fps budget itself, so 36–46% of the frames exceed 33.3 ms. At 25 fps the picture is different: only 1.6–2.5% of frames exceed 40 ms, but the 99th percentile (41.3–44.1 ms) still does not meet the deadline. The detector takes 26 ms per frame and association 3–4% of the total. Because the latencies cluster at the 30 fps boundary, small differences in association cost change the miss rate visibly. The final CLASP, with a median association of 1.23 ms against 0.94 ms for the frozen version, misses 45.7% of the 30 fps deadlines against 35.8%; CLASP-lite misses 39.5%, ByteTrack 40.1% and OC-SORT 36.7%. TrackTrack, at 19.2 fps, misses 97% of the 40 ms deadlines. The pipeline uses 3.9 cores, almost entirely in the multi-threaded detector. (These rates are for MOT17 alone; the 28 fps of Table 4 include the denser MOT20 sequences.) System energy is the same within 3% for the five lightweight variants (1.90–1.96 J per frame, paired ratios 0.97–1.00 against the final CLASP) and 1.43× higher with TrackTrack. Peak working set is 0.46–0.50 GB for all trackers. With a CPU detector, then, the 37% tracker-level regression of Sect. 5.4 is invisible in system energy but measurable in the fraction of frames that miss a 30 fps deadline.

5.8 Cost of the association mechanisms

The accuracy ablation of the companion paper removes one mechanism of CLASP at a time; Table 7 measures what each mechanism *costs*, on the paired bench,

Table 4 Complete pipeline (W2): detector in the loop, 2 repeats per sequence (MOT17 train; YOLO11n also MOT20 train), idle-adjusted RAPL package energy. Frame latency (decoding + detection + tracking) is the median over runs of the per-run median, 95th and 99th percentile. HOTA: mean of the per-dataset TrackEval combined scores of those sequences

			Frame latency (ms)					
Detector	Tracker	FPS	median	p95	p99	Tracker ms	Net J/fr.	HOTA
YOLO11m	CLASP	7.19	134.6	144.2	155.9	1.42	7.25	41.44
YOLO11m	ByteTrack	7.18	134.1	144.0	156.9	0.97	7.27	39.30
YOLO11m	OC-SORT	7.22	133.6	143.6	157.1	1.09	7.20	37.98
YOLO11m	TrackTrack	6.36	152.7	163.6	175.9	18.75	7.81	41.93
YOLO11m	DeepSORT	2.88	308.1	377.4	418.1	173.89	15.64	38.96
YOLO11s	CLASP	16.39	59.9	65.2	74.9	1.43	3.05	40.28
YOLO11s	ByteTrack	16.39	59.7	65.0	75.8	0.97	3.01	38.14
YOLO11s	OC-SORT	16.45	59.7	65.1	73.0	1.09	3.04	37.78
YOLO11s	TrackTrack	12.59	79.6	85.6	95.6	18.97	3.69	40.89
YOLO11n	CLASP	27.97	33.9	37.7	42.4	1.37	1.67	27.15
YOLO11n	ByteTrack	27.58	33.7	37.5	42.2	1.24	1.66	25.69
YOLO11n	OC-SORT	27.94	33.9	37.7	42.8	1.09	1.65	23.83
YOLO11n	TrackTrack	18.47	53.4	57.9	63.8	18.74	2.37	28.98

Table 5 Detector scale: measured detector cost (RAPL package; 36 sequences, 39 755 frames) and tracker share of system energy (detector + tracker-level energy of Table 1)

	ms/	J/	Tracker share of system energy			
Detector	frame	frame	CLASP	ByteTr.	OC-SORT	TrackTr.
YOLO11n	32.9	1.91	1.4%	1.0%	1.0%	24.3%
YOLO11s	61.2	3.71	0.7%	0.5%	0.5%	14.2%
YOLO11m	158.5	9.82	0.3%	0.2%	0.2%	5.9%

with all variants run through the reference code path so that differences reflect the mechanisms rather than the exact-computation layer. Two mechanisms are expensive: the appearance term on ambiguous pairs (41% of the reference energy) and the size-adaptive gate (a fixed 50 px gate admits far fewer candidate pairs and saves 42%, at a cost of 3.3 HOTA). Two design alternatives are *more* expensive than CLASP: velocity extrapolation adds per-pair prediction (+53%) and removing the cascade lets every detection compete with every track (+43%). The acceptance threshold costs 9%; the components added in the final version (observation-centric history, track-aware initialisation, larger appearance weight) and the remaining mechanisms change energy by at most 7% each, none significantly. The certified implementation reaches 0.67 of the reference energy in this bench (0.64 in Table 1).

5.9 Accuracy–cost trade-off

Figure 5 places the five trackers in the accuracy–cost plane, with HOTA on the 68 held-out sequences. On this pooled measure OC-SORT dominates CLASP: it is 1.4 HOTA more accurate and 26% cheaper per frame at tracker level. CLASP lies on the efficient frontier only on MOT17, MOT20 and the dataset-balanced mean, and its additional cost is invisible at system level (Table 4). TrackTrack buys its accuracy with 23× the tracker energy of CLASP, and DeepSORT is dominated on both axes. On seven never-observed MOT15 sequences, evaluated blind after pre-registration in the companion paper, the picture is less favourable. OC-SORT (53.6 HOTA) and ByteTrack (50.8) are both more accurate than CLASP (48.6) *and* cheaper, so CLASP is dominated there. Its accuracy position is specific to the benchmark family on which it was developed; its cost measurements are not.

5.10 Scheduling the detector (RQ4)

Because the detector takes about 97% of system energy, only its scheduling can save energy substantially. With YOLO11m on every frame as reference we tested three families on the eight design sequences, first by replaying cached detections and then live, with the detectors in the loop and RAPL energy (Table 8; Supplementary Sect. S11).

Temporal mixing (YOLO11m on some frames, YOLO11n on the others) behaves as a linear interpolation between the two detectors: YOLO11m on every second frame loses 3.3 HOTA for a 40% saving, and a frame-selection oracle is no better. Pre-registered frame skipping for the frozen CLASP lost 2.0–6.8 HOTA on held-out sequences, except skipping at most two frames after an unambiguous frame (−0.16 HOTA, 8.8% saving): skipping shifts the frames at which crossings are resolved.

Spatial verification keeps a decision on every frame: YOLO11n runs on the whole frame and YOLO11m on

Table 6 Per-frame deadline compliance, cores, energy and memory of the complete pipeline with YOLO11n in the loop (MOT17 train, 7 sequences, 2 repeats, all frames pooled). >40 / >33: percentage of frames exceeding 40 ms (25 fps) and 33.3 ms (30 fps). Assoc.: association latency p50/p99. Cores: process CPU time / wall time. J/fr.: RAPL package energy per frame (gross); ratio paired by block against the final CLASP. GB: peak working set

		Frame latency (ms)			Frames missing (%)						
Tracker	fps	p50	p95	p99	>40	>33	Assoc. (ms)	Cores	J/fr.	Ratio	GB
CLASP (final)	29.8	33.1	38.0	44.1	2.5	45.7	1.23/3.24	3.90	1.908	1.000	0.49
frozen CLASP	30.2	32.6	37.2	41.7	1.6	35.8	0.94/2.67	3.90	1.900	0.973	0.49
CLASP-lite	30.1	32.8	37.4	41.5	1.6	39.5	1.17/3.14	3.89	1.962	1.001	0.49
ByteTrack	30.0	32.8	37.7	42.5	2.2	40.1	0.86/1.44	3.89	1.906	0.994	0.46
OC-SORT	30.1	32.7	37.1	41.3	1.6	36.7	0.95/1.64	3.90	1.916	0.983	0.50
TrackTrack	19.2	53.1	58.1	61.5	97.4	100.0	20.04/24.50	3.89	2.855	1.425	0.47

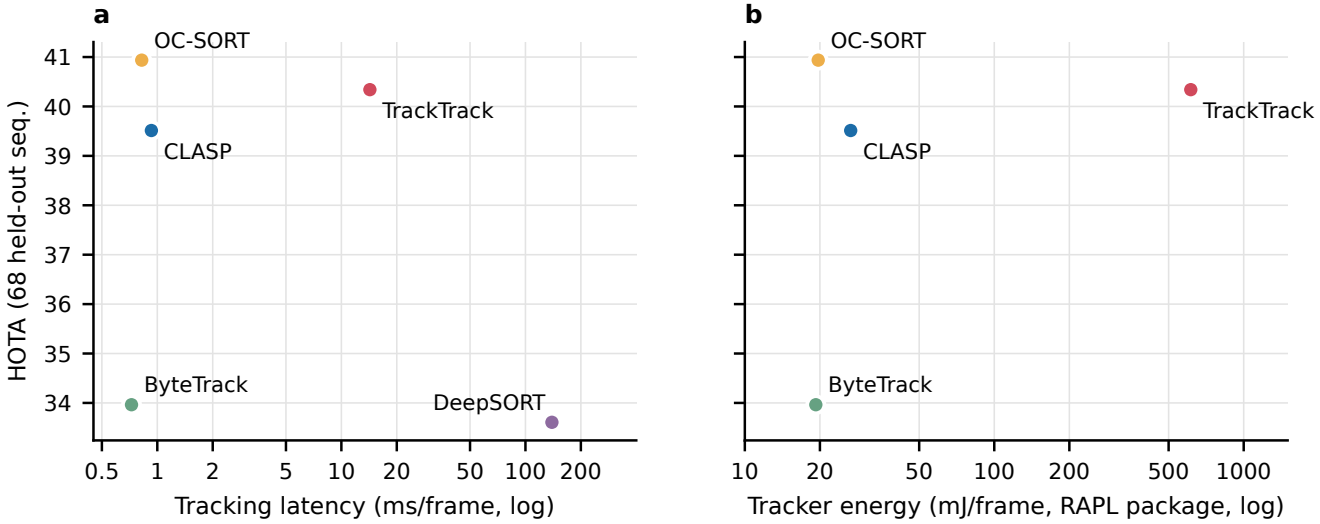


Fig. 5 Accuracy versus cost (YOLO11m). **a** Tracking latency in whole-sequence runs (76 sequences). **b** Tracker-level energy from the paired bench. HOTA: mean per sequence on the 68 held-out sequences. Log scales

Table 7 Cost of each mechanism of the final CLASP: paired bench (8 sequences × 300 frames, 5 rounds × 3 passes), reference code path. Energy ratio relative to the full model with 95% bootstrap CI over sequences; Wilcoxon p

Variant	ms/fr.	mJ/fr.	Ratio [95% CI]	p
Full CLASP (ref. path)	1.43	40.1	1.00	–
Velocity extrapolation	2.27	61.0	1.53 [1.34, 1.74]	0.008
No state cascade	2.10	56.4	1.43 [1.29, 1.60]	0.008
No track-aware init.	1.44	40.0	1.02 [0.97, 1.06]	1.000
Greedy instead of LAP	1.43	39.9	1.01 [0.96, 1.06]	0.945
No duplicate rescue	1.43	39.1	1.00 [0.96, 1.03]	0.547
Fixed admission thr.	1.42	38.8	0.99 [0.94, 1.04]	0.312
CLASP-lite	1.40	37.4	0.99 [0.95, 1.04]	0.148
No obs.-centric history	1.42	38.1	0.98 [0.94, 1.01]	0.148
No identity inheritance	1.43	39.0	0.97 [0.93, 1.00]	0.547
No confidence hysteresis	1.39	37.6	0.96 [0.91, 1.02]	0.148
No lost-memory recovery	1.39	37.9	0.96 [0.91, 1.02]	0.109
Appearance weight 0.3	1.36	36.9	0.93 [0.87, 0.99]	0.109
No acceptance threshold	1.31	35.4	0.91 [0.88, 0.96]	0.008
No ambiguity appearance	0.86	24.1	0.59 [0.48, 0.74]	0.008
Fixed gate (50 px)	0.83	22.3	0.58 [0.49, 0.68]	0.008
CLASP certified	1.00	27.2	0.67 [0.59, 0.73]	0.008

crops around selected boxes, packed into one canvas at the full-frame scale so that one call serves all crops (a call on a canvas of area a costs $0.57 + 3.5 \times 10^{-5} a$ J). A rule that refines every region the tracker knows about, in the spirit of CaTDet [18] and of time-constrained ROI detection [19], kept or slightly improved accuracy and saved 21–24%. A value-of-information rule—verify a box when the probability of a wrong decision without the expensive detector, $\min(p, 1-p)$, exceeds λ times its crop energy, with p learned online from the verified boxes—saved 34–41% at equal HOTA but lost 3.4–4.7 MOTA. Neither rule dominates. Its pre-registered held-out confirmation on 24 further sequences, run live, *failed*: in crowds the mosaic became larger than the frame (no cost cap was specified), which tripled energy on MOT20 while raising HOTA by 7 points, and on two DanceTrack sequences the online calibration became over-confident and stopped verifying.

Three platform lessons generalise. Replaying crop inference from full-frame caches is optimistic (YOLO11m on crops lost 4–6 recall points, with a sequence-dependent precision), so crop policies must be judged live; with Windows power throttling active YOLO11m took 280 instead of 147 ms; and the Ultralytics pre-

Table 8 Scheduling the detector (reference: YOLO11m on every frame with CLASP). Saving: energy per frame, from the measured per-call costs (replay) or measured with RAPL (live); Δ: mean per sequence. Panel: 8 design sequences; held-out: pre-registered confirmation on 24 sequences

Policy	Eval.	Save %	ΔHOTA	ΔIDF1	ΔMOTA
YOLO11n on every frame	replay, panel	80	-4.96	-5.44	-9.91
YOLO11m every 2nd frame, n otherwise	replay, panel	40	-3.29	-3.36	-4.32
VoI crop verification, $\lambda = 0.02$	live, panel	34	+0.59	+1.73	-3.44
VoI crop verification, $\lambda = 0.04$	live, panel	41	-0.59	-0.26	-4.74
Threshold verification (CaTDet-style), $\tau = 0.3$	live, panel	21	+0.70	+1.29	+0.03
Threshold verification, $\tau = 0.6$	live, panel	24	+0.37	+0.42	+0.10
VoI, $\lambda = 0.02$, pre-registered	live, 24 held-out	-2	-0.33	+0.71	+0.33

dictor stalled for up to 1.2 s on small inputs that the network processes in 25 ms, so the network was called directly with identical pre- and post-processing.

6 Discussion

Exactness as a design principle, and its limit. Lossy efficiency methods must be re-validated on every dataset and detector; exact ones do not, and the savings of E1, E2, E4 and T hold at every operating point. The certificate is different: its yield depends on how much the expensive term can move the decision. The corollary turns this into a design rule—if $w_a \geq c_{\max}$, do not try to certify—and the yield curve is a cheap, exact diagnostic of how often an appearance cue is decision-relevant at a given weight. The regression of Sect. 5.4 is the same fact seen from the cost side. Changing an accuracy parameter can silently remove an efficiency mechanism. Efficiency must therefore be re-measured, under the same paired protocol, every time a model is revised for accuracy.

Measure, do not count. Vectorisation, an unproductive certificate and the crop-inference replay (Sect. 5.10) all show that operation counts, micro-benchmarks and replays can disagree with measured energy. Paired designs with interleaved idle windows were necessary: background package power varied between 24 and 37 W over a day.

When does tracker efficiency matter? With YOLO11m a lightweight tracker is below 0.5% of system energy, and association savings—including the 36% of the exact layer—are invisible at that level; we state this plainly. They matter (i) with small detectors on edge CPUs, where a GMC tracker raises measured system energy by 43% and lowers throughput from 28 to 18.5 fps; (ii) in multi-camera or high-frame-rate settings, where association scales with tracks and detections while detection can be batched or offloaded; and (iii) when choosing among trackers, since GMC and deep ReID multiply system energy by 1.08–2.2×.

Spending the association budget on accuracy. The same cost structure can be used the other way round. Because association is about 1% of a YOLO11m frame, five copies of CLASP with different confidence thresholds can run on the same detections for 0.9% more system energy, and choosing among them per sequence with a label-free score raised HOTA by 1.2–1.4 and IDF1 by about 2.5 in a pre-registered confirmation [2].

Real time. No complete pipeline with YOLO11m reaches 10 fps on this CPU, and association is never the bottleneck for the lightweight trackers (p99 $\leq$ 2 ms). Real-time claims for CPU-only MOT should state the workload, the deadline and the fraction of frames that miss it: even with YOLO11n, 36–46% of frames miss a 30 fps deadline (Sect. 5.7), and a heavy tracker forfeits that operating point.

Generality. The certificate applies to any tracker whose cost is a geometric term plus a bounded, additive, expensive term (colour, texture, a motion likelihood or a learned descriptor with a known range); its yield is governed by the ratio of the bound to the acceptance threshold.

7 Limitations

Energy is measured on one CPU platform through RAPL package counters, which estimate processor energy, not wall power; absolute numbers will differ on ARM edge devices and with a calibrated external meter, while the paired relative results should transfer better. Temperature was not logged. ByteTrack and TrackTrack are reimplementations; no deep-ReID variants other than DeepSORT were measured, and DeepSORT was measured in an earlier campaign than the other trackers under the same protocol. The detector-scale shares of Table 5 combine separately measured detector and tracker energies; the end-to-end rows of Table 4 test their trend but cover MOT17 (and MOT20 for YOLO11n) only. Detector scheduling was designed on eight sequences and confirmed (unsuccessfully) on 24; spatial verification was measured with one detector pair. The accuracy figures that place costs in context come from the companion paper. There, the accuracy advantage of CLASP over ByteTrack measured on the benchmark family on which it was developed did not hold on never-observed sequences. The cost conclusions of this paper do not depend on that ranking, but any

accuracy–cost trade-off drawn from Fig. 5 is specific to MOT17, MOT20 and DanceTrack.

8 Conclusions

We measured, rather than estimated, what multi-person tracking costs on a CPU. Lightweight trackers meet 30 fps association deadlines on one core with margins above 16× and differ by at most 1.3% in system energy at every detector scale, while camera-motion compensation raises system energy by 8% with YOLO11m and by 43% with YOLO11n, and deep re-identification doubles it. Only the smallest detector brings a complete CPU pipeline near 30 fps on average, an operating point a heavy tracker cannot reach, and no tracker meets that deadline on every frame. The association of CLASP was made 36% cheaper without changing a single output, verified on 158 sequences. A dominance certificate halves the energy when the appearance cue is weak, and we proved and measured that it can never succeed once the appearance weight reaches the acceptance threshold—the setting at which appearance actually improves accuracy. Revising CLASP for accuracy therefore cost 37% more tracker energy. The certificate is therefore also an exact diagnostic of when an expensive cue is decision-relevant. Vectorisation increased energy, an unproductive certificate check is pure overhead, temporal detector scheduling lost accuracy, and spatial verification saved energy on design sequences but did not survive a pre-registered held-out test. Exact, certified computation skipping complements the approximate methods that dominate efficient tracking, within limits that can be predicted.

Acknowledgements The author thanks the maintainers of MOTChallenge and DanceTrack for the public benchmarks, and the authors of OC-SORT and TrackEval for releasing their code.

Statements and Declarations

Funding The author declares that no funds, grants or other support were received during the preparation of this manuscript.

Competing interests The author has no relevant financial or non-financial interests to disclose.

Data availability MOT17, MOT20 and DanceTrack are publicly available from their maintainers. The measurement pipeline, tracker code, detection caches, per-run RAPL telemetry, run manifest with SHA-256 hashes, pre-registrations and the scripts that regenerate and verify every table (`verify_tables.py`, `REPRODUCE.md`) are in a reproducibility archive that is available from the author on reasonable request and will be deposited in a public repository with a DOI upon acceptance. Dataset frames and ground truth are not redistributed.

Code availability See Data availability.

Ethics approval Not applicable: public benchmark datasets only; no new data from human participants.

Author contributions Single-author work.

Use of AI tools The author used a large language model (Claude, Anthropic) as an assistive tool, under the author's direction, for drafting and refactoring measurement and analysis code and for language editing. All measurements, results and claims were verified by the author, who takes full responsibility for the content.

References

1. Aharon, N., Orfaig, R., Bobrovsky, B.Z.: BoT-SORT: Robust associations multi-pedestrian tracking. arXiv:2206.14651 (2022). URL `https://arxiv.org/abs/2206.14651`
2. Baquero Delgado, S.: Choosing a tracker's operating point without labels: a near-free tracker bank for real-time multi-person tracking on CPUs (2026). Companion manuscript, submitted to Journal of Real-Time Image Processing
3. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: Proc. IEEE International Conference on Image Processing (ICIP), pp. 3464–3468 (2016). DOI 10.1109/ICIP.2016.7533003
4. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM, Philadelphia (2009). DOI 10.1137/1.9780898717754
5. Cao, J., Pang, J., Weng, X., Khirodkar, R., Kitani, K.: Observation-centric SORT: Rethinking SORT for robust multi-object tracking. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9686–9696 (2023). DOI 10.1109/CVPR52729.2023.00934
6. Crouse, D.F.: On implementing 2D rectangular assignment algorithms. IEEE Transactions on Aerospace and Electronic Systems **52**(4), 1679–1696 (2016). DOI 10.1109/TAES.2016.140952
7. David, H., Gorbatov, E., Hanebutte, U.R., Khanna, R., Le, C.: RAPL: Memory power estimation and capping. In: Proc. ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), pp. 189–194 (2010). DOI 10.1145/1840845.1840883
8. Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., Meng, H.: StrongSORT: Make DeepSORT great again. IEEE Transactions on Multimedia **25**, 8725–8737 (2023). DOI 10.1109/TMM.2023.3240881

9. Hähnel, M., Döbel, B., Völp, M., Härtig, H.: Measuring energy consumption for short code paths using RAPL. ACM SIGMETRICS Performance Evaluation Review **40**(3), 13–17 (2012). DOI 10.1145/2425248.2425252
10. Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., Pineau, J.: Towards the systematic reporting of the energy and carbon footprints of machine learning. Journal of Machine Learning Research **21**(248), 1–43 (2020). URL `http://jmlr.org/papers/v21/20-312.html`
11. Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing **38**(4), 325–340 (1987). DOI 10.1007/BF02278710
12. Kailath, T.: The divergence and Bhattacharyya distance measures in signal selection. IEEE Transactions on Communication Technology **15**(1), 52–60 (1967). DOI 10.1109/TCOM.1967.1089532
13. Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of Basic Engineering **82**(1), 35–45 (1960). DOI 10.1115/1.3662552
14. Khan, K.N., Hirki, M., Niemi, T., Nurminen, J.K., Ou, Z.: RAPL in action: Experiences in using RAPL for power measurements. ACM Transactions on Modeling and Performance Evaluation of Computing Systems **3**(2), 9:1–9:26 (2018). DOI 10.1145/3177754
15. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1–2), 83–97 (1955). DOI 10.1002/nav.3800020109
16. Luiten, J., Ošep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., Leibe, B.: HOTA: A higher order metric for evaluating multi-object tracking. International Journal of Computer Vision **129**, 548–578 (2021). DOI 10.1007/s11263-020-01375-2
17. Maggiolino, G., Ahmad, A., Cao, J., Kitani, K.: Deep OC-SORT: Multi-pedestrian tracking by adaptive re-identification. In: Proc. IEEE International Conference on Image Processing (ICIP) (2023). DOI 10.1109/ICIP49359.2023.10222576
18. Mao, H., Kong, T., Dally, W.J.: CaTDet: Cascaded tracked detector for efficient object detection from video. In: Proceedings of Machine Learning and Systems (MLSys), vol. 1 (2019). URL `https://proceedings.mlsys.org/paper_files/paper/2019/hash/146b291bfc50fb43464b8ef8b1fea5f0-Abstract.html`
19. Mitzel, D., Sudowe, P., Leibe, B.: Real-time multi-person tracking with time-constrained detection. In: Proceedings of the British Machine Vision Conference (BMVC), pp. 104.1–104.11 (2011). DOI 10.5244/C.25.104
20. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.M., Rothchild, D., So, D., Texier, M., Dean, J.: Carbon emissions and large neural network training. arXiv:2104.10350 (2021). URL `https://arxiv.org/abs/2104.10350`
21. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018). DOI 10.1109/CVPR.2018.00474
22. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green AI. Communications of the ACM **63**(12), 54–63 (2020). DOI 10.1145/3381831
23. Shi, J., Tomasi, C.: Good features to track. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 593–600 (1994). DOI 10.1109/CVPR.1994.323794
24. Shim, K., Ko, K., Yang, Y., Kim, C.: Focusing on tracks for online multi-object tracking. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2025). DOI 10.1109/CVPR52734.2025.01091
25. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Proc. 57th Annual Meeting of the Association for Computational Linguistics, pp. 3645–3650 (2019). DOI 10.18653/v1/P19-1355
26. Swain, M.J., Ballard, D.H.: Color indexing. International Journal of Computer Vision **7**(1), 11–32 (1991). DOI 10.1007/BF00130487
27. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking. In: Computer Vision – ECCV 2020, LNCS 12356, pp. 107–122. Springer (2020). DOI 10.1007/978-3-030-58621-8_7
28. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945). DOI 10.2307/3001968
29. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: Proc. IEEE International Conference on Image Processing (ICIP), pp. 3645–3649 (2017). DOI 10.1109/ICIP.2017.8296962
30. Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., Wei, Y.: MOTR: End-to-end multiple-object tracking with transformer. In: Computer Vision – ECCV 2022, LNCS 13687. Springer (2022). DOI 10.1007/978-3-031-19812-0_38
31. Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., Wang, X.: ByteTrack: Multi-object tracking by associating every detection box. In: Computer Vision – ECCV 2022, LNCS 13682, pp. 1–21. Springer (2022). DOI 10.1007/978-3-031-20047-2_1
32. Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W.: FairMOT: On the fairness of detection and re-identification in multiple object tracking. International Journal of Computer Vision **129**, 3069–3087 (2021). DOI 10.1007/s11263-021-01513-4
33. Zhang, Y., Wang, T., Zhang, X.: MOTRv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023). DOI 10.1109/CVPR52729.2023.02112

Biography

Santiago Baquero Delgado is with Universidad de La Salle, Bogotá, Colombia. Research interests: real-time multi-object tracking, energy measurement of vision pipelines and exact acceleration of data association on CPU-only hardware.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- supplementary.pdf