Multi-Object State Estimation using Probabilistic Belief-Based Trackers

Connecting Low-Frequency Detection and High-Rate Prediction on Embedded Devices

Vitalii Mashkov

Supervisor(s): Qing Wang, Nitish Kumar

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Vitalii Mashkov
Final project course: CSE3000 Research Project
Thesis committee: Qing Wang, Nitish Kumar, Braden Refalo

Abstract

Agents require object-centric world models to enable Active Inference, where decisions minimize 'surprise'. Maintaining high-frequency state estimation on edge hardware presents a dilemma between detection accuracy and update frequency. Traditional tracking frameworks are designed for high-frequency data and fail to bridge the large spatial uncertainty gaps that accumulate during low-frequency detection. We propose a Probabilistic Belief Tracker that decouples high-frequency belief propagation from low-frequency perception. The system utilizes a Gaussian Sum Filter with Interacting Multiple Model inspired dynamics to maintain competing motion hypotheses, representing multi-modal spatial uncertainty during detection gaps. Our results demonstrate that switching to these probabilistic beliefs provides the high-frequency continuity and identity stability required by a reliable world model. Deployment on the NVIDIA Jetson Nano confirms the architecture is viable for real-time edge deployment, while MOT17 benchmarks show that using 6x fewer detections (5 FPS) drops tracking accuracy by 10.7% and identity stability by 9.2%, relative to the 30 FPS baseline. Limiting identity switches to 172 at a 5 FPS detection rate confirms that probabilistic continuity preserves identity stability, despite the reduction in overall tracking accuracy typical of sparse detections.

1 Introduction

To interact effectively with the world, an autonomous agent must move beyond processing a raw stream of pixels and instead maintain a persistent, object-centric internal map. This representation provides the probabilistic foundation for Active Inference, where an agent minimizes its variational free energy by evaluating a generative internal model against sensory evidence [1; 2]. Traditional point-estimates cause "surprise" (free energy) spikes during missed detections, because they disregard state uncertainty. In contrast, a continuous, uncertainty-aware probabilistic belief maintains the temporal smoothness required for stable planning. Bridging this gap requires an Active Inference-compatible perception layer that represents and estimates spatial uncertainty during observational gaps; otherwise, target loss or identity swaps break the agent's internal map and lead to planning failures.

Multi-Object Tracking (MOT) transforms low-level detections into higher-level world understanding. On resource-constrained edge hardware, this process faces a critical trade-off between detection accuracy and update frequency [3]. While using lightweight detectors allows for high frame rates, it often sacrifices detection accuracy, which results in unstable position estimates and false negatives. Conversely, high-accuracy models typically force the system into a low-frequency mode, introducing latency in the world model. Traditional trackers struggle in the second scenario, as they assume high-frequency observations [4], an assumption that fails under the low detection rates of edge hardware. Moreover, as identified by Cao et al. [5], the standard linear motion models underlying these frameworks suffer from severe error accumulation when deprived of continuous observations. This makes them unable to bridge the spatial uncertainty gaps created by infrequent updates or detector latency, which leads to frequent identity switches (IDSW). We evaluate these constraints on a low-power embedded platform, specifically the NVIDIA Jetson Nano. Throughout this work, we use the terms *edge*, *embedded*, and *resource-constrained* interchangeably to describe this hardware context.

Our approach resolves these challenges by decoupling state estimation from detection, ensuring that the agent maintains the continuous world model required for Active Inference, regardless of the detector's operational frequency. We propose a Probabilistic Belief Tracker that separates high-frequency belief propagation from the lower-frequency correction cycle that integrates detections and visual embeddings. To prevent the "surprise" caused by point estimates during detector latency or occlusions, we model the state as a mixture of multiple motion hypotheses that maintain competing beliefs. This preserves tracking continuity and accounts for non-linear movement during observational gaps until new detections arrive.

This leads us to our research question: **To what extent can a dual-rate, asynchronous belief-based architecture maintain object-state continuity and identity consistency under sparse detections, satisfying the perception requirements for Active Inference agents on resource-constrained hardware?** We propose that by explicitly accounting for uncertainty during detection gaps through multi-hypothesis state estimation, we minimize identity switches (IDSW) and keep tracks alive even when the sensor data is sparse or noisy.

We have made three research contributions in this work:

1. A dual-rate asynchronous tracking architecture that decouples belief propagation from detection latency, bypassing the synchronous bottleneck that limits standard synchronous trackers on edge hardware.
2. A probabilistic belief representation that maintains target identity across detection gaps, modeling object states as a mixture of competing motion hypotheses instead of a single trajectory estimate.
3. An empirical validation on standard MOT benchmarks and NVIDIA Jetson Nano hardware, demonstrating that maintaining continuous probabilistic beliefs provides a stable, identity-consistent world state suitable for resource-constrained systems.

Section 2 builds the theoretical groundwork for belief-based estimation, while Section 3 details the core architecture of the Probabilistic Belief Tracker. Following the experimental setup and benchmarks in Section 4, we provide our empirical results in Section 5, where we evaluate the system's resilience to sparse detections. Section 6 provides a discussion of these findings, including an analysis of failure cases and hardware constraints. Finally, we conclude with potential directions for future work (Section 7) and address the ethical and responsible use of AI in this research (Section 8).

2 Background & Related Work

Multi-Object Tracking (MOT) serves as a foundation for representing a dynamic environment. Trackers link detections across consecutive frames to estimate object trajectories and maintain identity consistency. Modern tracking frameworks use the 'tracking-by-detection' approach, where a detector identifies targets and a motion model tracks their state over time. Motion models usually rely on recursive Bayesian filtering, such as the Kalman Filter (KF), combined with association criteria such as Intersection-over-Union (IoU) or visual similarity.

2.1 Requirements for Active Inference Perception

Active Inference agents operate by minimizing variational free energy through a continuous cycle of perception and action. This requires a generative internal model that can predict future states and resolve sensory uncertainty [1; 2]. Combining these principles with real-time state estimation constraints, we define three specific requirements:

1. **Temporal Belief Continuity:** A continuous, high-frequency probability distribution (belief) of the world state must be maintained, rather than relying on frame-bound updates to satisfy control loop demands.
2. **Identity Stability:** The internal model must maintain consistent object identities across observational gaps to support long-term planning and avoid "surprise" spikes caused by track fragmentation.
3. **Latency Compensation:** Delayed detections must be retroactively realigned with the belief timeline to ensure planning is based on the estimated current state, preventing prediction errors from stale data.

Although these requirements are foundational to the Active Inference framework, standard tracking systems rely on dense, high-frequency detections to stabilize their simple unimodal motion models. This dependency cannot be satisfied on resource-constrained edge hardware without sacrificing detection accuracy.

2.2 Related Work

Tracking research originally focused on simple spatial matching, but modern work has moved toward deep-learning models either for visual feature extraction or for end-to-end tracking. SORT [4] paired constant-velocity motion with IoU matching, which the DeepSORT [6] framework later extended with visual embeddings to preserve identity during occlusions. More recent frameworks like ByteTrack [7], OC-SORT [5], and BoT-SORT [8] prioritize using low-confidence detections, accounting for non-linear motion, and compensating for camera movement. While these frameworks rely on the linear Kalman Filter (KF) [9], their reliance on a single Gaussian posterior limits them to a unimodal representation. The effectiveness of this unimodal estimation depends on high-frequency updates; the linear model fails as sampling intervals increase, since infrequent detections often violate the constant-velocity assumption.

When detections are infrequent, an object's potential path is no longer a single point; it becomes a set of discrete possibilities, such as continuing straight, braking, or turning. A unimodal estimate accounts for these distinct modes by scaling the state covariance $\mathbf{P}$. This causes the search gate to expand and overlap with nearby objects, including neighboring candidates, leading to identity switches once the target reappears. Gaussian Sum Filtering (GSF) [10] addresses this by representing the state as a sum of Gaussians, while Interacting Multiple Model (IMM) systems [11] provide the theoretical foundation for representing these competing hypotheses.

Going beyond traditional filtering, research has transitioned to end-to-end Transformer architectures like MOTR [12] and TrackFormer [13]. However, replacing manual motion heuristics with learned motion patterns results in significant computational overhead. For instance, MOTR achieves only 7.5 FPS on an NVIDIA V100 [12], making real-time edge deployment impractical. Furthermore, for frameworks that rely on visual appearance, observational gaps often introduce motion blur and dramatic pose changes that degrade visual embeddings. On embedded hardware, the computational cost of extracting visual embeddings often creates a further trade-off between model complexity and update frequency.

To address these spatial uncertainties and temporal latencies on edge hardware, the tracking system requires a decoupled, multi-hypothesis architecture.

2.3 The Out-of-Sequence Measurement (OOSM) Problem

Asynchronous architectures by design encounter the Out-of-Sequence Measurement (OOSM) problem [14; 15], where detector delay creates a temporal mismatch between the arriving measurement $\mathbf{z}_{t-k}$ and the current belief $\mathbf{x}_t$. Applying delayed data directly to the current belief introduces localization errors and causes track fragmentation, as the measurement belongs to a past state. Resolving these errors necessitates a retroactive realignment of the track's history. Analytical shortcuts exist for multi-step lags [15], but we rely on a Rollback-and-Replay (RR) approach (Algorithm 3, detailed in Section 3) to maintain the GSF mixture integrity, providing an accurate world state required for Active Inference planning.

3 Methodology: Probabilistic Belief Tracker

The Probabilistic Belief Tracker maintains a continuous world model for agents operating with sparse or delayed detections. Standard trackers rely on frequent unimodal updates; our architecture instead models object states using continuous multi-modal probability distributions. The system maintains the high-frequency continuity and identity stability required for Active Inference, even during long observational gaps, by modeling spatial uncertainty through multiple motion hypotheses.

3.1 Overview

The Probabilistic Belief Tracker addresses the trade-off between detection lag and tracking stability using a dual-rate asynchronous architecture. Figure 1 shows how this tracker is designed to interface with two asynchronous threads that share a common belief history, using mutexes to prevent concurrent execution.

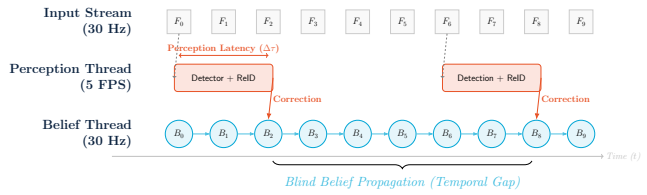

Figure 1: Dual-rate asynchronous tracking pipeline. The 30 Hz belief thread maintains temporal continuity while the 5 Hz perception thread provides periodic sensory corrections.

1. Belief Thread (High-Frequency): This thread runs at the camera's source frequency, typically 30 Hz. It manages spatial uncertainty while propagating object states to generate continuous estimates required for smooth state transitions. While 30 Hz is used for benchmarks, the thread supports arbitrary rates.
2. Perception Thread (Low-Frequency): This thread operates in the background to execute computationally intensive upstream tasks, such as object detection and feature extraction to generate the sensory observations for the tracker (updates). We use a 5 FPS baseline to represent the typical throughput of high-accuracy models on edge hardware (based on hardware profiling in Section 5.5), but the architecture is frequency-independent and remains stable even under variable delays or irregular detector updates.

Decoupling these rates keeps the agent's world model synchronized with the current frame. This configuration provides the predictive state extrapolation required to bridge the data gaps where standard synchronous trackers typically fail.

3.2 Probabilistic Belief Representation

The belief of each tracked object is represented as a tuple $B = (X, A)$. Here, X denotes a kinematic mixture modeled by a GSF [10], while A represents an appearance state tracked via an Incremental Gaussian Mixture Model (IGMM) in the feature space (detailed in Section 3.4). Kinematic state X consists of a mixture of n Gaussian components. We define the state of an individual component i as an 8D vector:

$$\mathbf{x}_i = [cx, cy, a, h, v_{cx}, v_{cy}, v_a, v_h]^T \tag{1}$$

tracking the bounding box center (cx, cy), aspect ratio a, height h, and the respective velocities $(v_{cx}, v_{cy}, v_a, v_h)$.

Following the heuristics in DeepSORT [6], the initial state covariance P_0 is scaled proportionally to the target height h to account for varying object scales:

$$\sigma_p = \frac{h}{20}, \quad \sigma_v = \frac{h}{160}, \quad \sigma_a = 0.1 \tag{2}$$

$$P_{pos} = \text{diag}([2\sigma_p, 2\sigma_p, \sigma_a, 2\sigma_p])^2 \tag{3}$$

$$P_{vel} = \text{diag}([10\sigma_v, 10\sigma_v, 10^{-4}, 10\sigma_v])^2 \tag{4}$$

where σ_a is a constant aspect ratio noise term, preventing premature gating failures during silhouette or pose changes, while other spatial uncertainty scales with target's physical size.

3.3 Motion Dynamics and Prediction

The tracker propagates each component on every source frame using a multi-model formulation [11]. We employ a damped constant-velocity model where positional components (cx, cy, a, h) are incremented by their respective velocities scaled by Δt, while velocity components decay by an effective damping factor $d = \delta^{\Delta t}$. To limit linear extrapolation errors during missed detections, we use the $\delta = 0.90$ damping factor from StrongSORT [16]. We further set the base damping factors δ of 0.50 for braking (DECEL) and 0.85 for turns. The TURN model includes a constant-rate pre-rotation of the velocity vector by $\omega\Delta t$ (where $\omega = 0.5$ rad/s) before the state transition matrix $\mathbf{F}$ is applied, representing a standard pedestrian motion prior. This linear approximation avoids the complexity of a full coordinated turn model but remains accurate at high frame rates. For larger temporal gaps, the system integrates the state using 50 ms numerical sub-steps to maintain the stability and accuracy of the uncertainty estimates during long extrapolations.

We approximate motion uncertainty (process noise) using height-scaled discrete white noise. We adjust the position and velocity noise terms using a model-specific factor Q_{scale} (CV $= 1.0$, DECEL $= 0.8$, $\text{TURN}_{\text{L/R}} = 1.5$) and a speed-adaptive multiplier $k_{\text{scale}} = Q_{\text{scale}} \cdot (1.0 + 0.005 \cdot \|\mathbf{v}\|)$, while keeping aspect ratio noise terms constant:

$$\mathbf{q}_{\text{pos}} = [k_{\text{scale}}\sigma_p, k_{\text{scale}}\sigma_p, 0.01, k_{\text{scale}}\sigma_p] \tag{5}$$

$$\mathbf{q}_{\text{vel}} = [k_{\text{scale}}\sigma_v, k_{\text{scale}}\sigma_v, 10^{-5}, k_{\text{scale}}\sigma_v] \tag{6}$$

$$Q_{\text{final}} = \text{diag}([\mathbf{q}_{\text{pos}}, \mathbf{q}_{\text{vel}}])^2 \cdot \Delta t \tag{7}$$

where $\|\mathbf{v}\|$ is the absolute pixel velocity. This allows uncertainty to scale faster during non-linear turns and high-speed maneuvers. We used a grid search on the MOT17 train set to determine the hyperparameters, optimizing for identity stability (IDF1) as the primary continuous measure for tracking consistency (since the upstream detector and embeddings are fixed). Algorithm 1 defines the complete belief prediction cycle.

Algorithm 1 Belief Prediction Loop

```
Input: Target frame index k, Active belief registry R
Compute temporal step Δt since last prediction
if Δt > 0 then
    for each active belief B_j ∈ R do
        Propagate belief components via motion dynamics ▷ (Sec 3.3)
    end for
end if
Output: Active beliefs
```

3.4 Observation Model and Hybrid Association

When new detections arrive, the tracker performs a multi-stage association process using a hybrid likelihood metric. We evaluate the likelihood of matching a detection against the kinematic belief X and the appearance state A. The visual likelihood L_{igmm} is evaluated using an Incremental Gaussian Mixture Model (IGMM), which maintains a history of appearance embeddings, while a rolling average momentum

feature provides a motion-based likelihood L_{mom}. We combine these into a joint appearance-momentum likelihood L_{app} to stabilize target identities prior to the GSF state update. To integrate these terms, we employ a time-adaptive weighting strategy:

$$L_{\text{app}} = (1 - w_{\text{mom}}) \cdot L_{\text{igmm}} + w_{\text{mom}} \cdot L_{\text{mom}} \tag{8}$$

where $w_{\text{mom}} = \min(0.8, 0.3{+}0.1{\cdot}t_{\text{since\_update}})$, with $t_{\text{since\_update}}$ measured in frames. Increasing w_{mom} during tracking gaps accounts for rapid appearance drift while maintaining the stable appearance baseline. Algorithm 2 details this component-level measurement update sequence.

To narrow the spatial search and adjust filtering sensitivity under high uncertainty, we apply Velocity Tracking Consistency (VTC) scaling to the velocity sub-covariance during likelihood evaluation: $P_{\text{vel}} \cdot (0.6 + 0.4 \cdot \max(0, \cos\_\text{sim}))$, where cos_sim is the velocity cosine similarity. Furthermore, an innovation scale factor (s_{scale}) inflates the innovation covariance matrix $\mathbf{S}$ if recent prediction errors are large:

$$s_{\text{scale}} = 0.96 \cdot s_{\text{scale}} + 0.04 \cdot \max(1.0, \frac{d_M^2}{4}) \tag{9}$$

where d_M^2 is the squared Mahalanobis distance and the innovation scale is bounded such that $s_{\text{scale}} \in [0.5, 4.0]$ to ensure numerical stability during noisy detection sequences. This makes the filter more cautious when observations deviate significantly from predictions.

Algorithm 2 GSF Measurement Update and Splitting

Input: GSF Mixture components C, Detections D, Appearance Model M, Miss Probability p_{miss}
$C_{temp} \leftarrow \emptyset$
for each component $c \in C$ **do**
  $C_{miss} \leftarrow \text{SpeedAdaptiveSplit}(c)$ ▷ Generate CV, Decel, and Turn hypotheses for miss branches
  for each component $c_m \in C_{miss}$ **do**
    $w_m \leftarrow w_m \cdot p_{miss}$
    $C_{temp} \leftarrow C_{temp} \cup \{c_m\}$
  end for
  for each detection $\mathbf{z}_j \in D$ **do**
    if $\mathbf{z}_j$ passes spatial gating checks for c **then**
      $\mathbf{x}', \mathbf{P}' \leftarrow \text{KalmanUpdate}(c, \mathbf{z}_j)$ ▷ Correct state with OCM velocity stabilization
      $w' \leftarrow \text{ComputeWeight}(c, \mathbf{z}_j, M)$ ▷ Combine spatial and appearance likelihoods
      $C_{temp} \leftarrow C_{temp} \cup \{(w', \mathbf{x}', \mathbf{P}')\}$
    end if
  end for
end for
Output: Expanded candidate components C_{temp}

3.5 Rollback-and-Replay (RR) Mechanism

To resolve detection latency from frame k in a dual-rate system, we employ the RR mechanism detailed in Algorithm 3, which mirrors the optimal chronological reprocessing standard established by Bar-Shalom [14; 15]. This mechanism resolves temporal misalignment by retroactively realigning late-arriving detections with the high-frequency belief timeline. This ensures the 30 Hz control loop relies on current, synchronized data rather than outdated sensor input. The system resolves detection latency by rolling back the belief history and re-propagating the states in chronological order.

Algorithm 3 Belief Update and Realignment

Input: Frame index k, Detections $\mathbf{D}_k$, Registry $\mathcal{R}$
$\mathcal{H}_{dets}[k] \leftarrow \mathbf{D}_k$ ▷ Registry detection log
if $k \leq k_{last}$ **then** ▷ Delayed Detection (OOSM)
  $k_{live} \leftarrow k_{last}$
  Rollback Registry history to frame $k-1$ ▷ Rewind registry snapshots
  Predict forward to frame k
  Update matched beliefs at k with $\mathbf{D}_k$ ▷ Two-Stage Assoc.
  for $t = k+1$ **to** k_{live} **do** ▷ Replay Fast-Forward
    Predict all beliefs to frame t
    Update beliefs at t using $\mathcal{H}_{dets}[t]$
  end for
  $k_{last}, k_{predict} \leftarrow k_{live}$
else ▷ In-Order Update
  Predict all beliefs forward to frame k
  Update matched beliefs at k with $\mathbf{D}_k$
  $k_{last}, k_{predict} \leftarrow k$
end if
$\mathcal{R}.prune\_histories(k)$ ▷ Archive old snapshots
Output: Updated belief registry $\mathcal{R}$

The system keeps a 2.0-second history buffer; if a detection arrives with a delay beyond this window, we discard the data and terminate the belief to avoid introducing drift. Upon completing the fast-forward phase, we normalize the adaptive covariance multipliers by 50% to restore the filter's baseline sensitivity. This reset ensures that the heightened uncertainty from the realignment process does not cause the system to ignore subsequent physical observations.

3.6 Hypothesis Management and Belief Lifecycle

We manage the GSF components lifecycle using Certainty-based Management (CBM) and Speed-Adaptive Splitting (SAS), which merge and branch hypotheses. On every update cycle, the SAS mechanism generates alternative motion hypotheses (weighted by $P_{\text{miss}} = 0.08$) representing the missed-detection branch. If the target speed exceeds 90 px/s, SAS branches these hypotheses to represent alternative models. To reduce overhead, we disable turn branching for slow-moving targets; trajectory curvature only significantly affects association accuracy at higher velocities. The CBM module merges redundant hypotheses if their Mahalanobis distance in the 4D measurement space (cx, cy, a, h) falls below 1.2, using the base component's innovation covariance to assess similarity. CBM also prunes components with weights $w < 0.02$ to maintain efficiency. Algorithm 4 outlines the component lifecycle control, including weighted merging and pruning.

The Gaussian Sum Filter provides the continuous, uncertainty-aware probability distribution required for Active Inference agents to minimize surprise during observational gaps. This yields a belief confidence score (Eq. 10) for scenarios where point-estimate trackers fail to capture the multimodal uncertainty of non-linear motion.

Each belief outputs a certainty score:

$$c = c_{\text{cov}} \cdot c_{\text{weight}} \tag{10}$$

Algorithm 4 GSF Mixture Lifecycle Management

Input: Candidate components C_{temp}
$C_{pruned} \leftarrow \{c \in C_{temp} \mid w_c \geq 0.02\}$ ▷ Prune low-probability components
$C_{merged} \leftarrow \text{MergeSpatiallyClose}(C_{pruned}, \text{threshold} = 1.2)$ ▷ Merge via Mahalanobis distance
$C_{final} \leftarrow \text{NormalizeTopComponents}(C_{merged}, n_{\max} = 4)$ ▷ Enforce cardinality constraint
Output: Normalized GSF components mixture C_{final}

where $c_{\text{weight}} = 1 - \left(-\sum_{i=1}^{n} w_i \ln(w_i + 10^{-9})\right) / \ln(n)$ represents the normalized entropy-based weight concentration of the mixture (for $n > 1$ components, with $c_{\text{weight}} = 1.0$ otherwise) and c_{cov} captures the spatial uncertainty contraction:

$$c_{\text{cov}} = \exp\left(-0.5 \cdot \left(\frac{\det(P_{\text{pos},t})}{\det(P_{\text{pos},t_{\text{det}}})} - 1.0\right)\right) \tag{11}$$

where $P_{\text{pos},t}$ is the current positional covariance and $P_{\text{pos},t_{\text{det}}}$ is the covariance at the last confirmed detection. When the weight concentration reaches 1.0, the belief has converged on a single motion hypothesis, resolving the 'surprise' of competing estimates. We suppress output for any belief where $c < 0.02$ to eliminate unstable ghost tracks generated by high-velocity prediction errors. To ensure stable belief formation, we utilize a dual-threshold birth strategy: detections with confidence ≥ 0.6 initialize a belief, while those ≥ 0.8 are confirmed immediately to bypass the initial hits requirement. Finally, we terminate beliefs that remain undetected for more than 1.0 second.

4 Experimental Setup

We test the tracker's resilience to sparse and delayed updates using a setup that specifies the evaluation datasets, baseline trackers, and hardware deployment platforms.

4.1 Datasets and Metrics

While our tracker is class-agnostic, we evaluate its performance on pedestrian tracking using the MOT17 [17] benchmark to facilitate comparison with standard baselines. Due to the privacy of the official test labels, each training sequence was split into two halves (split_idx $= \lfloor \text{seqLength}/2 \rfloor$), where the second half was used for validation. A detailed breakdown of the unique challenges in each sequence, including camera positioning, crowd density, and lighting conditions, is provided in Table A.3 (Appendix).

We compute all metrics using the TrackEval framework [18]. Our primary evaluation criteria are Higher Order Tracking Accuracy (HOTA) [19], ID F1 Score (IDF1), and total ID Switches (IDSW), as they directly reflect identity stability, which is essential for maintaining a reliable world model in Active Inference. We also include Multiple Object Tracking Accuracy (MOTA) for historical context and comparison. By using the same detections and visual embeddings across all models, we show that the improvements in Section 5 are due to our tracker's logic.

4.2 Baseline Selection and Evaluation Protocol

We compare the Probabilistic Belief Tracker against five standard baselines: StrongSORT [16], ByteTrack [7], DeepSORT [6], OC-SORT [5], and SORT [4].

Benchmarks use a 30 FPS detection rate to establish a performance baseline in high-frequency environments. We isolate tracking performance from detector noise and localization error by using precomputed YOLOX [20] detections. For trackers that require visual features, we use ResNeSt50 [21] embeddings to provide a uniform appearance baseline. We adopt the precomputed detection and embedding from StrongSORT [16], following their protocol. We disable offline pre- and post-processing, such as trajectory interpolation and smoothing, to ensure the benchmarks reflect real-time performance on edge hardware. Our hardware profiling (Section 5.5) shows that the Jetson Nano achieves a perception throughput of 5 FPS with a 152 ms detector latency (equivalent to a 5-frame delay). Accordingly, we select 5 FPS and a 5-frame delay as the baseline operational parameters for the sensitivity sweeps. This setup allows for a direct comparison with the state-of-the-art tracker (Section 5.1) before we investigate its resilience to the sparse data conditions common on edge hardware (Section 5.2). Baseline sensitivity to detection frequency (Section 5.2) was evaluated using two methods: (1) a track-retention approach, where the most recent tracks are carried over to all intermediate frames, and (2) an empty-detection protocol, where intermediate frames receive no new data. Although empty detections cause immediate track fragmentation in all unimodal baselines, we adopt this protocol for our primary benchmarks to simulate sparse update conditions. The alternative track-retention approach is evaluated in Appendix B.2.

4.3 Hardware and Implementation Details

We use two platforms to evaluate the tracker on both algorithmic accuracy and practical edge performance. Baseline benchmarks for MOT17 were run on a workstation with an M4 Pro processor and 24 GB of RAM, allowing us to evaluate the performance free from hardware bottlenecks.

To test the system on resource-constrained hardware, we deploy it on an NVIDIA Jetson Nano (2 GB). We use the native NVIDIA DeepStream SDK with Python bindings (`pyds`) to implement the dual-rate architecture. Table A.2 (Appendix) lists the specific JetPack and TensorRT versions used. To maintain stable performance within a 10 W power budget, we optimize the system for the Jetson Nano through:

- **Model Quantization:** The YOLOv26n detector and OSNet-x0_25 [22] appearance model are compiled into TensorRT engines with FP16 precision.
- **System Tuning:** We lock the Jetson Nano in its maximum performance mode (MAXN, 10 W) and disable the desktop GUI to free up available unified memory.
- **Asynchronous Execution:** The `pyds` interface allows the perception models and the belief propagation thread to run on independent cycles, to prevent detection latency from blocking state updates.

We use ResNeSt50 for baseline accuracy comparisons (Section 4.2), but we switch to the lighter OSNet-x0_25 for edge deployment, which is optimized for person re-identification. The hyperparameters and core settings used across all experiments are detailed in Table A.1 (Appendix).

5 Results

The experimental results show that the tracker maintains identity stability and world state continuity under strict computational constraints. We evaluated the system against standard baselines, performed frequency and latency sensitivity sweeps, conducted component ablation studies, and profiled real-time execution on edge hardware.

5.1 Comparative Evaluation on MOT17

We evaluated the Probabilistic Belief Tracker against five standard baselines on the MOT17 validation set using a standard tracking-by-detection paradigm (1:1 detection-to-update ratio) to establish a baseline under dense updates. As shown in Table 1, StrongSORT achieves the highest HOTA, yet our tracker yields the highest identity stability and lowest identity fragmentation, reducing IDSW by 53.3% relative to StrongSORT and by 70.3% relative to SORT. This 53.3% reduction in identity switches is primarily due to the integration of observation-centric momentum (OCM) and mixture-based filtering, which prevents identity drift during occlusions. Table B.1 (Appendix) provides a detailed sequence-specific breakdown.

Table 1: Tracking performance on MOT17 validation (1:1 detection-to-update ratio).

Method	HOTA ↑	IDF1 ↑	MOTA ↑	IDSW ↓
StrongSORT	**65.84**	77.99	**72.80**	253
ByteTrack	63.41	74.39	72.49	314
DeepSORT	62.05	71.17	72.01	315
OC-SORT	61.24	71.40	70.54	289
SORT	59.57	66.78	70.59	398
Belief Tracker (Ours)	64.51	**78.43**	72.79	**118**

5.2 Sensitivity Analysis of Detection Frequency

We evaluated tracking stability through a simulated frequency sweep from 30 FPS to 1 FPS. While Figures 3 and 4 report global metrics, Figure 2 offers a qualitative look at how frequency affects tracking. The sequence shows every tenth frame after a belief update, comparing the belief propagation in the 1 FPS configuration (top) against the dense updates of the 30 FPS.

Figure 2: Visual comparison on sequence MOT17-04: the 1 FPS configuration (top) maintains identity across significant spatial gaps, while the 30 FPS configuration (bottom) reflects standard high-frequency update density.

As shown in Figure 3, overall tracking accuracy (HOTA) for our tracker degrades as detections become more infrequent, falling from 64.67 to 37.12 at the 1 FPS limit. However, identity stability remains consistent; at the 5 FPS target rate (our edge baselines), the IDSW count is 172. Even at the extreme 1 FPS limit, the system maintains 157 IDSW.

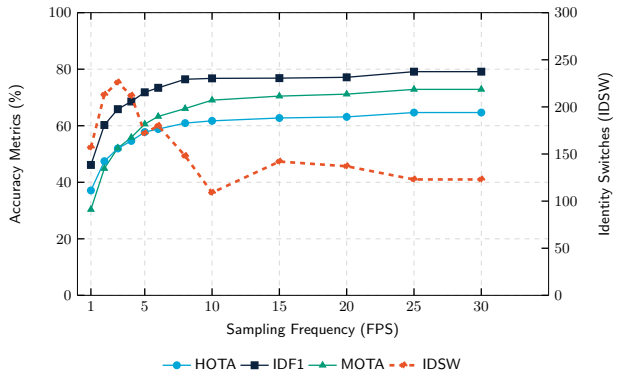

Figure 3: Sensitivity of the Probabilistic Belief Tracker to detection frequency (30–1 FPS) on the MOT17 validation set. Multi-hypothesis tracking maintains identity consistency even as spatial precision (HOTA) degrades.

Although HOTA decreases for all models as detections become sparser, our tracker maintains 57.77 HOTA at 5 FPS, whereas all synchronous baselines drop to 0.00 HOTA under the empty-detection protocol due to immediate track fragmentation (Figure 4, Appendix Table B.3). To establish a performance upper bound for the baselines, we also evaluated them under a track-retention protocol (Appendix Table B.2), in which the last known tracks are repeated. Despite this protocol advantage, the baselines experience significant track instability at 5 FPS: our tracker records only 172 identity switches (IDSW), compared to 249 and 574 for StrongSORT and ByteTrack, respectively. Under the empty-detection protocol, our decoupled architecture maintains track continuity at lower frequencies where synchronous baselines fail to preserve active trajectories.

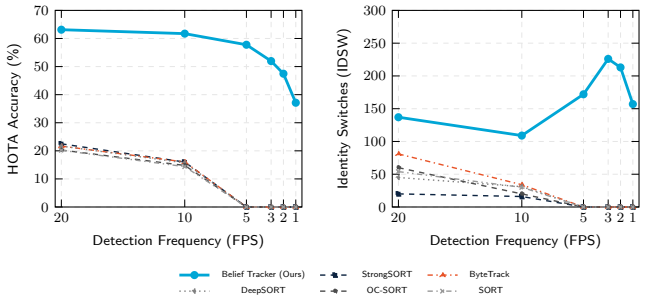

Figure 4: Comparative sensitivity of the Probabilistic Belief Tracker and synchronous baselines to detection frequency (20–1 FPS) on the MOT17 validation set. Multi-hypothesis tracking preserves identity stability in sparse environments where unimodal baselines fragment.

5.3 Sensitivity Analysis of Detector Latency

We tested asynchronous realignment by delaying detections at the 5 FPS target rate, sweeping from zero to 20 frames (up to 666 ms), with 5 frames representing our edge delay. As lag increases, HOTA and IDF1 drop due to the uncertainty added during fast-forwarding (Table 2). However, identity switches decrease as the delay grows, from 172 down to 96 at the 20-frame limit. This occurs because the RR mechanism (Algorithm 3) accumulates prediction uncertainty during the replay phase; rather than making incorrect associations across long gaps, the tracker defaults to starting new trajectories. While MOTA drops by 50% at 20 frames, the rollback overhead is negligible, at only 0.29 ms per update. Full results are in Table B.5 (Appendix).

Table 2: Sensitivity of the Probabilistic Belief Tracker to detector lag (5 FPS, MOT17).

Delay (Frames)	HOTA ↑	IDF1 ↑	MOTA ↑	IDSW ↓	Latency ↓
0 (Baseline)	57.77	71.83	60.59	172	2.16 ms
5 (166 ms)	51.13	63.84	47.44	190	2.13 ms
10 (333 ms)	46.47	57.21	40.13	161	2.21 ms
20 (666 ms)	39.17	48.21	29.79	96	2.45 ms

5.4 Component Ablation Study

The ablation results in Table 3 show the contribution of each module at 30 FPS. Removing the Appearance Model (APP) has the most severe impact on identity consistency: IDSW more than triples (from 118 to 395) and latency rises to 13.05 ms. This latency increase results from a combinatorial expansion of motion hypotheses: without visual features to filter unlikely matches, the number of mixture components grows rapidly, which outweighs the savings of skipping appearance comparisons. Certainty-based Management (CBM) is critical for precision, without which MOTA drops by 8.63%. Observation-centric Momentum (OCM) contributes 3.9% to the overall HOTA score. Speed-Adaptive Splitting (SAS) improves tracking continuity (HOTA +0.76%, MOTA +0.56%) at the cost of only one additional identity switch. The RR mechanism has no impact on these 30 FPS metrics, as it is designed specifically for the high-latency gaps analyzed in Section 5.2.

Table 3: Ablation study of the Probabilistic Belief Tracker (30 FPS, MOT17).

Configuration	HOTA ↑	IDF1 ↑	IDSW ↓	MOTA ↑	Latency ↓
Full Architecture	**64.51**	**78.43**	**118**	**72.79**	**9.79 ms**
No Appearance (APP)	62.03	74.21	395	71.91	13.05 ms
No Certainty (CBM)	61.29	75.15	124	64.16	9.58 ms
No Momentum (OCM)	60.60	75.14	150	65.93	9.54 ms
No Speed-Adaptive (SAS)	63.75	77.88	117	72.23	9.13 ms
No Rollback (RR)	64.51	78.43	118	72.79	9.17 ms

5.5 Hardware Deployment Profiling

We profiled the tracker on an NVIDIA Jetson Nano (10 W MAXN) under a three-object indoor tracking scenario to simulate an average load. The results show that the dual-rate architecture successfully distributes the computational load within embedded hardware constraints.

The Perception Thread operates at roughly 81.5% GPU utilization during object detection and feature extraction, with a detector latency of 152.06 ms. Because the combined throughput of detection and feature extraction limits the Perception Thread to 5 FPS, the Belief Thread operates asynchronously on the CPU, bypassing the bottleneck. The multimodal GSF propagation requires only 0.17 ms per frame for `predict` and 0.39 ms for asynchronous `update`. This efficiency enables the Belief Thread to maintain a 30 Hz world model at 72.3% overall CPU load, with memory stabilizing at 2.13 GB. These benchmarks indicate that our tracking architecture is suitable for real-time edge deployment, as it provides high-frequency estimates within the thermal and compute budgets of autonomous systems.

6 Discussion

Tracking on resource-constrained hardware forces compromises between mathematical precision and execution speed, creating trade-offs that we analyze here alongside failure cases and camera motion limits.

6.1 Theoretical Implications of Belief-Based Tracking

Transitioning from deterministic tracking requires balancing algorithmic complexity with hardware limits. We now analyze how these trade-offs influence world state stability.

Identity stability under high uncertainty follows a non-monotonic trend, where extreme degradation (20-frame delay, 96 IDSW; 1 FPS, 157 IDSW) reduces identity switches compared to moderate peaks (5-frame delay, 190 IDSW; 3 FPS, 226 IDSW). This occurs because both forward prediction during sparse updates and retroactive realignment during latency inflation increase the spatial uncertainty of the belief. Instead of establishing a tentative association with a highly uncertain belief, which risks identity swaps, the tracker suppresses or terminates the belief. While this preserves the world model's integrity by avoiding false certainties, it leads to the significant recall loss reflected in our MOTA scores. In this context, IDSW acts more as a measure of system honesty than tracking precision; the tracker would rather lose an object than provide a corrupted history that could cause planning failures.

Kinematic gating and branching restrict turn-hypothesis branching to targets faster than 90 px/s to control computational load. The 0.56% MOTA decrease without SAS (Section 5.4) shows that turn-branching for fast-moving targets preserves trajectory continuity at 30 Hz under real-time execution constraints.

By maintaining a single, linear timeline instead of parallel trajectories, chronological realignment for OOSM ensures world-state synchronization without the memory overhead associated with managing divergent trajectory branches. The 0% impact observed without this mechanism at 30 FPS (Section 5.4) demonstrates that the RR mechanism is a specialized corrective measure, required only when high sensor latency would introduce temporal misalignment.

Adaptive identity anchoring shifts association weight toward the rolling appearance momentum feature as observational gaps grow. This prevents identity drift, which com-

monly occurs when visual embeddings degrade due to pose variations or partial occlusions. This appearance anchor matches re-emerging objects against their running average representation established during the trajectory instead of static embeddings. Separately, the 3.9% HOTA decrease when removing OCM (Section 5.4) shows that integrating velocity history protects against kinematic drift.

The 70.3% reduction in identity switches compared to SORT (Table 1) is primarily due to our multi-hypothesis GSF framework. Deterministic trackers often orphan a target if its motion violates the constant-velocity assumption. Consequently, the next detection falls outside the singular Mahalanobis search gate. Maintaining parallel motion components allows our tracker to recover from non-linear maneuvers that would typically trigger track resets. This multi-hypothesis resilience preserves the world state during the observational gaps common in low-frequency observations.

6.2 Hardware Trade-offs and Embedded Constraints

We prioritize world state continuity over high-frequency detections to remain within the Jetson Nano's 10 W power budget. Profiling (Section 5.5) demonstrates that CPU-bound propagation (`predict`) requires only 0.17 ms, while retroactive updates (`update`) take 0.39 ms. This efficiency allows the belief thread to propagate at 30 Hz, while the low update latency allows the system to integrate delayed 5 FPS detections without interrupting the real-time loop. By decoupling these threads, the system provides the temporal continuity required for Active Inference without exceeding the embedded platform's CPU limits.

6.3 Empirical Failure Cases and Limitations

The dual-rate architecture improves baseline stability, but specific conditions expose the theoretical limits of our current association logic.

Sequence MOT17-04 reveals the limits of the association logic in dense environments (see Table A.3, Appendix). When targets maneuver close together, their GSF spatial uncertainty ellipses overlap. This produces near-identical likelihoods for single detections. If visual embeddings are also degraded by occlusion, the matching algorithm minimizes the global cost, which can result in a locally incorrect assignment. This overlap accounts for the 23 identity switches recorded for MOT17-04, one of the higher totals among the evaluated sequences (Table B.1, Appendix).

Low-light environments, particularly the MOT17-10 night sequence (see Appendix), highlight the risks of starting beliefs without a confidence-based quality threshold. Despite the appearance model's ability to extract unique identity embeddings, any inaccuracy in the first detection persists throughout the belief's lifecycle because the system initializes its baseline target identity embedding from an object's first raw detection. This becomes critical during observational gaps, as the adaptive weighting mechanism prioritizes this potentially noisy reference to maintain the belief. In dark scenarios, the tracker effectively locks onto a noisy identity. These noisy identity locks lead to association failures that the GSF cannot resolve.

Despite these results, the MOT17 dataset may oversimplify the real-world challenges of mobile robotics due to its predictable, smooth camera paths. In practice, autonomous platforms experience nonlinear trajectories and rapid camera jitter. We omitted Camera Motion Compensation (CMC) to remain within the Jetson Nano's compute budget. While methods like ECC-based alignment [23] in StrongSORT [16] or faster ORB matching [24] as seen in BoT-SORT [8] exist, they introduce significant overhead. Without corrections, camera shifts during detection gaps cause prediction drift. This introduces a vulnerability where a single disturbance during the 152 ms detector lag can push the belief outside the search gate before visual confirmation arrives. Overcoming this limitation requires integrating low-latency ego-motion estimation directly within the high-frequency belief thread. Finally, hardware profiling was restricted to three active objects; evaluating resource scaling in crowded environments is left for future work.

7 Conclusion and Future Work

Decoupling belief propagation from perception bypasses the computational bottlenecks that restrict edge tracking performance, resolving the trade-off between detection accuracy and world state continuity. Our results show that continuous probabilistic beliefs preserve HOTA and IDF1 even under sparse detections. Specifically, a $6\times$ reduction in detection frequency (from 30 FPS to 5 FPS) maintains belief continuity, whereas all synchronous baselines drop to 0.00 HOTA. These outcomes demonstrate that multi-hypothesis continuity preserves identity consistency during observational gaps. Finally, deployment on the NVIDIA Jetson Nano confirms that the 30 Hz world model remains within a 10 W budget, providing the temporal and efficiency foundation required for Active Inference planning.

Building on the current perception layer, we identify three research paths to mitigate the limitations identified in our evaluation. First, incorporating learnable motion priors would capture common movement patterns in specific scenes. Biasing transition models toward these frequent paths would improve prediction accuracy and narrow the spatial search area during long occlusions (Section 6.3). Second, a confidence-gated approach would strengthen identity stability. Rather than locking the target identity embedding on the first detection, the tracker should wait for a high-quality observation or average the features over several frames. This prevents anchoring to blurry or dark images, which was identified as a primary cause of identity drift in Section 6.3. Third, integrating IMU data would improve reliability on mobile platforms. While standard Camera Motion Compensation (CMC) methods are often too compute-intensive for embedded devices, an IMU-based approach offers a lighter alternative for correcting the rapid camera jitter noted in Section 6.3, hence stabilizing target coordinate predictions and preventing planning failures caused by camera motion.

8 Responsible Research and Use of AI

Deploying real-time tracking systems on edge devices introduces broader social concerns, from surveillance ethics and carbon footprint to reproducibility and our use of AI tools.

8.1 Ethical Considerations and Dual-Use

Although the MOT17 [17] benchmark requires privacy measures like face-blurring, tracking ethics involve more than dataset rules. Any system that can maintain identity through long gaps is inherently dual-use; a tracker that helps a robot navigate a crowd can just as easily be used for unauthorized surveillance.

Our design attempts to mitigate this through 'privacy-by-design'. By reducing the perception rate to 5 FPS, we limit the density of biometric data captured by the system. While 30 Hz systems can resolve high-frequency identifiers like behavioral movement patterns, a 5 FPS stream makes such data difficult to extract. Processing data locally on the edge platform (Section 4.3) also keeps raw video off centralized servers. That said, this is no replacement for strict rules on data usage, and the tracking failures observed in complex scenarios (Section 6) demonstrate that vision alone remains insufficient for safety-critical deployment.

8.2 Sustainability and Environmental Impact

Our research prioritizes environmental sustainability by optimizing state estimation for low-power edge devices. Decoupling perception from belief propagation reduces the utilization of the GPU, as the most compute-intensive models only run at 5 FPS. This architecture supports long-term autonomous operation and eliminates the energy overhead of high-frequency cloud processing or high-power GPU clusters. Our evaluation (Section 4.3) shows that this efficiency enables robust tracking performance entirely within the power constraints of an embedded device, offering a path to more sustainable and energy-independent robotic systems.

8.3 Reproducibility

The tracker source code, NVIDIA Jetson demonstration environment along with deployment scripts and pre-trained quantized models are available via GitHub [25].

The specific mathematical parameters, such as the branching thresholds and damping factors, are summarized in Table A.1 (Appendix). Detailed information regarding the hardware configuration and software versions used during our evaluation is provided in Section 4.3.

8.4 Usage of AI and Large Language Models

In accordance with TU Delft's academic integrity guidelines, I disclose the use of Large Language Models (LLMs) during the development of this thesis. I used these models for structural guidance and correction of grammar, specifically to critique the paper's organization and readability. For the literature review, I have utilized LLM powered tools, such as NotebookLM for managing and summarizing related research in the necessary fields. I also used the tools for the hardware preparation, where they assisted in configuring the edge software environment and resolving dependency conflicts for the deployment scripts. All technical designs, experimental methods, and interpretations of results are my own original work. Each suggestion provided by the AI was reviewed for technical accuracy and manually refined before including in the final paper.

References

[1] Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. Active inference on discrete state-spaces: a synthesis. *Journal of Mathematical Psychology*, 99:102447, 2020.

[2] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 11525–11538, 2020.

[3] Jan Müller and Adrian Pigors. Efficient multi-object tracking on edge devices via reconstruction-based channel pruning. *arXiv preprint arXiv:2410.08769*, 2024.

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[5] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9686–9696, 2023.

[6] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[7] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[8] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[9] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[10] Daniel Alspach and Harold Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.

[11] Henk AP Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783, 1988.

[12] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Motr: End-to-end multiple-object tracking with transformer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 659–675. Springer, 2022.

[13] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8844–8854, 2022.

[14] Yaakov Bar-Shalom. Update with out-of-sequence measurements in tracking: exact solution. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):769–777, 2002.

[15] Yaakov Bar-Shalom, Huimin Chen, and Mahendra Mallick. One-step solution for the multistep out-of-sequence-measurement problem in tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 40(1):27–37, 2004.

[16] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 25:8725–8737, 2023.

[17] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[18] Jonathon Luiten and Arne Hoffhues. Trackeval. https://github.com/JonathonLuiten/TrackEval, 2020.

[19] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129:1851–1881, 2020.

[20] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[21] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2736–2746, 2022.

[22] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3702–3712, 2019.

[23] Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008.

[24] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[25] Vitalii Mashkov. Probabilistic belief tracker: Dual-rate asynchronous tracking and demo for edge devices. https://github.com/vitaliymsh/research-belief-tracker, 2026.

A Experimental Setup & Reproducibility Details

This appendix details the technical configurations and dataset parameters required to reproduce our experimental results.

A.1 System Configuration and Edge Specifications

Table A.1 lists the tracker's hyperparameters, and Table A.2 outlines the software environment deployed on the NVIDIA Jetson Nano.

Table A.1: Tracker hyperparameters.

Category	Parameter	Value
State	Dimension	8D
Init Cov	Pos/Vel/Aspect	$h/20, h/160, 0.25h$
Mixture	Max/Merge	4 / 1.2
	P_{miss} / v_{thresh}	0.08 / 90.0 px/s
Dynamics	Max Step	50 ms
Damping	CV/DEC/TURN	0.9 / 0.5 / 0.85
App.	Max / α_{igmm}	5 / 0.05
	w_{mom}	0.3–0.8
Lifecycle	Acq. Thresh	0.6 / 0.8
	Max Age/Hist	1.0 s / 2.0 s
Environment	Config	5 FPS / 30 Hz

Table A.2: Jetson Nano software environment.

Component	**Version**
OS	Ubuntu 18.04.6
JetPack	4.6.6
L4T	32.7.6
CUDA	10.2.300
cuDNN	8.2.1.32
TensorRT	8.2.1.9
OpenCV	4.1.1
DeepStream	6.0.1
Python	3.6.9

A.2 MOT17 Sequence Characteristics and Challenges

Table A.3 summarizes the MOT17 validation sequences. These sequences feature diverse challenges, such as camera motion, high crowd density, and poor lighting, which we use to evaluate the tracker in Section 4.

Table A.3: Characteristics of the MOT17 validation sequences.

Sequence	**Description**	**Core Challenges**	**Limitations**
MOT17-02	Static camera, eye-level plaza (Venice).	Baseline for appearance tracking and occlusion handling in moderate crowds.	Static setup lacks ego-motion or parallax challenges.
MOT17-04	Elevated static camera, busy street scene.	Tests small-target detection and stability in very dense environments.	Steep overhead angle distorts features, complicating ReID comparisons.
MOT17-05	Moving handheld camera, shopping street.	Resilience to erratic camera jitter, motion blur, and sharp viewpoint shifts.	Short duration limits assessment of long-term trajectory drift.
MOT17-09	Static camera, low angle, street corner.	Focuses on severe inter-object occlusion and significant scale changes.	Mostly linear motion; does not fully test complex maneuvering models.
MOT17-10	Moving camera, elevated, night scene.	Evaluates robustness in low-light and high sensor noise.	Extreme lighting often causes ReID failures in basic models.
MOT17-11	Moving camera, mall interior.	Tests non-linear paths and sustained occlusions in indoor crowds.	Long occlusions may exceed standard statistical gating limits.
MOT17-13	Moving camera (on a bus), busy intersection.	Tests heavy camera panning and complex, multi-directional flow.	Distance and fast transitions degrade visual embedding reliability.

B Extended Result Tables

This appendix lists the raw tracking metrics analyzed in Section 5.1.

B.1 Probabilistic Belief Tracker Detailed Performance Analysis

Table B.1 shows the tracking metrics for each sequence of the Probabilistic Belief Tracker under sparse detections.

Table B.1: Per-sequence results for the Probabilistic Belief Tracker on MOT17 validation.

Sequence	HOTA ↑	DetA ↑	AssA ↑	IDF1 ↑	MOTA ↑	FP ↓	FN ↓	IDSW ↓	MT ↑	ML ↓
MOT17-02-FRCNN	46.17	44.17	48.64	61.84	51.27	687	4097	47	16	10
MOT17-04-FRCNN	76.68	73.09	81.18	90.72	86.76	1113	2071	23	58	3
MOT17-05-FRCNN	48.21	51.13	45.81	55.13	62.15	177	1088	8	16	21
MOT17-09-FRCNN	58.15	62.07	54.59	73.51	76.69	46	620	8	12	2
MOT17-10-FRCNN	55.12	51.40	59.63	73.84	64.87	286	1785	18	14	4
MOT17-11-FRCNN	56.70	54.62	59.35	67.90	62.89	419	1252	9	18	14
MOT17-13-FRCNN	58.42	57.40	59.58	75.55	70.11	103	841	5	21	6
Overall	**64.51**	**60.80**	**69.34**	**78.43**	**72.79**	**2831**	**11754**	**118**	**155**	**60**

B.2 Baseline Sensitivity Analysis under Different Detection Protocols

We evaluate the sensitivity of the synchronous baselines to detection frequency under two simulation protocols. Table B.2 presents baseline performance under the *track-retention protocol*, which repeats the last known track state across intermediate frames. Table B.3 shows the corresponding performance under the *empty-detection protocol*, where intermediate frames receive no detections, to simulate highly sparse update conditions.

Table B.2: Baseline tracking performance on MOT17 under varying detection rates using the track-retention protocol.

		Detection Rate (FPS)					
Method	**Metric**	**20**	**10**	**5**	**3**	**2**	**1**
StrongSORT	**HOTA ↑**	62.24	60.24	53.96	49.40	43.82	33.19
	IDF1 ↑	74.44	72.45	65.46	61.11	54.06	39.53
	MOTA ↑	69.86	66.84	57.48	48.63	38.69	21.84
	IDSW ↓	232	291	249	216	188	103
ByteTrack	**HOTA ↑**	61.19	58.71	52.61	47.41	42.66	33.98
	IDF1 ↑	72.64	70.38	63.51	57.33	51.28	40.46
	MOTA ↑	70.09	66.70	58.92	50.09	41.91	26.55
	IDSW ↓	293	549	574	491	436	263
DeepSORT	**HOTA ↑**	59.08	56.59	50.58	45.56	40.49	31.04
	IDF1 ↑	68.66	66.52	60.66	56.43	49.85	36.83
	MOTA ↑	68.86	64.69	54.93	45.41	36.00	19.72
	IDSW ↓	278	373	320	236	206	102
OC-SORT	**HOTA ↑**	59.45	57.15	51.91	46.32	41.40	31.39
	IDF1 ↑	70.46	68.05	62.47	55.58	49.32	37.55
	MOTA ↑	67.41	64.49	55.86	46.93	38.97	24.06
	IDSW ↓	262	256	242	299	261	217
SORT	**HOTA ↑**	57.19	56.14	50.52	45.26	40.47	31.34
	IDF1 ↑	65.28	65.84	60.22	55.11	49.35	37.48
	MOTA ↑	67.25	63.46	54.37	45.53	37.16	22.39
	IDSW ↓	340	317	249	256	241	188

Table B.3: Baseline tracking performance on MOT17 under varying detection rates using the empty-detection protocol.

		Detection Rate (FPS)					
Method	**Metric**	**20**	**10**	**9**	**8**	**5**	**1**
SORT	**HOTA (%)**	20.20	14.37	0.11	0.11	0.00	0.00
	IDF1 (%)	14.12	7.40	0.02	0.02	0.00	0.00
	MOTA (%)	8.75	4.41	0.01	0.01	0.00	0.00
	IDSW	54	30	0	0	0	0
ByteTrack	**HOTA (%)**	21.67	16.03	0.00	0.00	0.00	0.00
	IDF1 (%)	15.41	8.55	0.00	0.00	0.00	0.00
	MOTA (%)	9.01	4.71	0.00	0.00	0.00	0.00
	IDSW	81	34	0	0	0	0
OC-SORT	**HOTA (%)**	20.28	14.80	0.11	0.11	0.00	0.00
	IDF1 (%)	14.29	7.74	0.02	0.02	0.00	0.00
	MOTA (%)	8.55	4.48	0.01	0.01	0.00	0.00
	IDSW	60	20	0	0	0	0
DeepSORT	**HOTA (%)**	21.69	15.71	0.00	0.00	0.00	0.00
	IDF1 (%)	15.31	8.28	0.00	0.00	0.00	0.00
	MOTA (%)	8.82	4.56	0.00	0.00	0.00	0.00
	IDSW	45	31	0	0	0	0
StrongSORT	**HOTA (%)**	22.50	16.00	0.00	0.00	0.00	0.00
	IDF1 (%)	16.00	8.26	0.00	0.00	0.00	0.00
	MOTA (%)	9.12	4.70	0.00	0.00	0.00	0.00
	IDSW	20	16	0	0	0	0

B.3 Probabilistic Belief Tracker Sensitivity Analysis under Frequency and Latency Sweeps

We evaluate the Probabilistic Belief Tracker across varying operational conditions. Table B.4 details the tracker's performance to sparse updates through a simulated frequency sweep from 30 to 1 FPS. Table B.5 evaluates the RR mechanism under simulated detector delays at a target rate of 5 FPS, with latencies ranging from 0 to 20 frames.

Table B.4: Sensitivity of the Probabilistic Belief Tracker to detection frequency (MOT17).

Rate (FPS)	HOTA ↑	IDF1 ↑	MOTA ↑	IDSW ↓	Update (ms)	Predict (ms)	Total (ms)
30	64.67	79.13	72.86	123	9.66	0.11	9.76
25	64.67	79.13	72.86	123	9.53	0.11	9.64
20	63.13	77.13	71.18	137	7.92	0.17	5.32
15	62.74	76.84	70.48	142	8.96	0.19	5.39
10	61.73	76.76	69.05	109	8.40	0.22	4.10
8	60.93	76.43	66.08	148	9.09	0.25	2.99
6	58.83	73.42	63.29	180	8.77	0.25	2.49
5	57.77	71.83	60.59	172	9.12	0.26	2.07
4	54.63	68.57	55.86	212	8.12	0.25	1.48
3	51.96	65.86	52.10	226	8.28	0.25	1.25
2	47.46	60.26	44.88	213	7.76	0.26	0.89
1	37.12	46.14	30.38	157	8.51	0.28	0.63

Table B.5: Sensitivity of the Probabilistic Belief Tracker to detector lag (MOT17).

Delay (frames)	HOTA (%)	IDF1 (%)	MOTA (%)	IDSW	Update (ms)	Predict (ms)	Total (ms)
0	57.77	71.83	60.59	172	9.54	0.27	2.16
1	56.50	70.44	58.07	166	9.08	0.30	2.10
2	55.13	69.12	55.04	180	9.06	0.33	2.12
3	53.47	67.39	52.51	164	9.42	0.33	2.18
4	52.11	65.37	49.86	174	9.02	0.33	2.11
5	51.13	63.84	47.44	190	9.07	0.36	2.13
6	50.17	62.08	45.48	176	9.13	0.36	2.14
7	49.20	61.35	43.84	159	9.11	0.37	2.15
8	48.40	59.75	42.14	163	9.16	0.41	2.19
9	47.60	58.76	40.99	182	9.19	0.41	2.20
10	46.47	57.21	40.13	161	9.19	0.43	2.21
12	44.94	55.34	37.98	146	9.31	0.47	2.26
14	43.96	54.36	35.82	140	9.51	0.52	2.34
16	41.95	51.73	33.51	115	9.53	0.55	2.36
18	39.02	48.02	31.21	112	9.75	0.58	2.43
20	39.17	48.21	29.79	96	9.64	0.63	2.45