Electronics and Information Technologies, 2026, 33, 131–144
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC 004.89

ENTROPY-GUIDED TRACKER SWITCHING METHOD FOR UNMANNED AERIAL VEHICLE REAL-TIME TRACKING

Volodymyr Oleksiuk**, **Serhiy Velhosh
Department of Radiophysics and Computer Technologies,
Ivan Franko National University of Lviv,
107 Gen. Tarnavskoho St., 79017 Lviv, Ukraine

Oleksiuk, V., Velhosh, S. (2026). Entropy-Guided Tracker Switching Method for Unmanned Aerial Vehicle Real-Time Tracking. *Electronics and Information Technologies*, 33, 131–144. https://doi.org/10.30970/eli.33.10

ABSTRACT

Background. Auto-guidance for unmanned aerial vehicles (UAVs) requires reliable real-time target tracking on resource-constrained onboard hardware. Modern state-of-the-art CNN-based and Transformer-based deep trackers provide strong accuracy but are often too slow and computationally expensive for continuous deployment on edge devices. In contrast, lightweight correlation-filter trackers run at high frame rates but can easily drift or lose the target because of occlusions or fast maneuvers. This robustness–efficiency trade-off (edge AI paradox) motivates adaptive strategies that balance accuracy, speed, and resource usage while preserving compute headroom for other onboard tasks.

Materials and methods. We propose an entropy-guided tracker switching method that combines a lightweight kernelized correlation filter (KCF) tracker augmented with Kalman motion prediction and a more accurate Siamese deep tracker. A motion-entropy scheduler quantifies the unpredictability of target motion using a normalized Shannon entropy over recent orientation changes. To avoid reacting to transient spikes, the entropy is exponentially smoothed, and threshold rules (with hysteresis) determine when KCF is sufficient and when to activate the deep tracker.

Results and Discussion. Experiments on UAV benchmarks (UAV123, OTB100) show that the hybrid tracker improves success AUC by ~10% over KCF and reaches about 70% of a Transformer tracker's AUC while running 1.5–3× faster than always-on deep tracking. The switcher invokes the deep tracker only during difficult intervals, sustaining real-time operation (~100 FPS) and reducing average computation to ≈0.6 GFLOPs per frame versus ≈1–4 GFLOPs for purely deep tracking.

Conclusion. The proposed motion-entropy scheduler enables an adaptive trade-off between efficiency, speed, and accuracy. It maintains high tracking precision during target maneuvers and occlusions by temporarily switching to a robust tracker yet saves computational load during steady-motion periods. This framework offers a practical solution for high-performance UAV tracking on the edge, while leaving resource headroom to apply other improvement techniques.

Keywords: object tracking; correlation filters; Siamese network; motion entropy; hybrid tracker; edge computing.

INTRODUCTION

Real-time target tracking by unmanned aerial vehicles (UAVs) demands both high accuracy and high efficiency. UAV onboard computers have limited processing power and energy yet must handle fast-moving targets and complex backgrounds. This creates a

trade-off often termed an edge AI paradox, where achieving high accuracy with a deep neural model conflicts with strict resource and real-time constraints [1].

On one hand, modern deep trackers (e.g., using Siamese networks or transformers) can achieve excellent accuracy on benchmarks, but their high computational cost makes real-time deployment challenging without powerful hardware. Recent transformer-based trackers such as TransT [2] provide strong accuracy, but their computational cost can still be challenging on resource-constrained UAV platforms; lightweight UAV-oriented trackers (e.g., MobileTrack [3]) were designed for high-speed deployment, which still use too many resources.

On the other hand, classical lightweight trackers like those based on correlation filters run at hundreds of frames per second on CPU, but tend to lose the target during occlusions, rapid appearance changes, or unexpected maneuvers [4, 5]. Ensuring accurate tracking despite such challenges while staying within the computational constraints is a key problem in UAV tracking.

Several factors make UAV tracking particularly difficult. The UAV's moving camera leads to a wide field of view with many distractors (e.g., ground clutter, other objects) and frequent viewpoint changes. Targets often occupy a small image region and can undergo extreme scale changes or become fully occluded by obstacles. Furthermore, wind or platform vibrations introduce motion blur and sudden target displacements [6]. These conditions can cause a simple tracker to drift off target or fail entirely. For example, a high-speed correlation filter tracker like KCF (Kernelized Correlation Filter) excels in nominal conditions with its Fast Fourier Transform (FFT) based template matching [4], but under occlusion or background confusion, it can quickly degrade [7]. Once the target is lost, a naive tracker cannot recover on its own [8]. Recent UAV systems also combine correlation filters with modern detectors to improve robustness under occlusion and scale changes [9, 10].

To address these issues, we propose a hybrid tracking architecture that leverages the complementary strengths of lightweight correlation-filter trackers and resource-intensive trackers. The core idea is to run a fast tracker most of the time and only switch to a computationally intensive but more robust one when necessary. Our framework uses KCF as a lightweight correlation-filter tracker for its real-time speed, augmented with a Kalman filter to predict target motion and smoothly update the region of interest (ROI). KCF's speed and good performance on stable motion make it a strong baseline [4]. However, when the target's motion becomes erratic, or the scene changes in a way likely to confuse KCF, we temporarily hand off to a resource-intensive tracker based on a Siamese neural network. The resource-intensive tracker (in our case, a variant of a MobileTrack Siamese model) is more computationally intensive but can handle significant appearance change and re-detect the object if it disappears and reappears. By combining these, we aim for "the best of both worlds": the efficiency of KCF during easy intervals and the resilience of a deep tracker during hard intervals.

A novel aspect of our approach is the motion entropy scheduler that decides when to switch between trackers. Instead of relying solely on the short-term tracker's internal confidence (e.g., correlation response strength via average peak-to-correlation energy (APCE) to detect failure [11, 12]) or activating the resource-intensive tracker on a fixed periodic verification trigger (e.g., every t frames) [13], we quantify the unpredictability of the target's recent motion using an entropy measure. Intuitively, when the target moves in a steady, predictable manner, a lightweight correlation-filter tracker should suffice. But when the motion is highly irregular or complex (high entropy), it likely foreshadows situations that could confuse the lightweight correlation-filter tracker (such as abrupt maneuvers or interacting objects). In those moments, engaging the more powerful resource-intensive tracker can prevent tracking loss. Our scheduler computes the Shannon entropy [14] of the target's motion orientation distribution and uses it as a switching signal. We introduce a Kalman-filtered motion model to estimate the target

trajectory and compute entropy in real time. High entropy values trigger a switch to the deep tracker, while low entropy allows a switch back to the fast tracker, with hysteresis to avoid rapid oscillations.

In addition, to explicitly handle UAV camera shake (that can cause constantly high entropy), we introduce a measure of residual flow. Intuitively, we first estimate the "smooth" global motion of the frame that corresponds to the drone's overall movement (for example, a rotation or steady flight), and then we look at the residual flow – the remaining pixel motion after subtracting this global shift. When the drone flies smoothly, this residual motion is well structured, and its entropy is low. When the platform is shaken by wind or aggressive maneuvers, the residual flow becomes chaotic, and its entropy increases sharply. We use this value as part of our frame reliability criteria: if the residual flow entropy is high, the frame is treated as unreliable, and the tracker skips online those corrupted frames.

MATERIALS AND METHODS

Physical Meaning and Theoretical Foundation of Motion Entropy

From a physical perspective, entropy is traditionally viewed as a measure of disorder or statistical uncertainty in a dynamical system. In the context of visual object tracking, motion entropy serves as a quantitative indicator of the trajectory's dynamical complexity and the degree of randomness in the target's displacement vector field. When an object moves uniformly and predictably, its displacement vectors exhibit a high degree of spatial correlation (concentrated in a narrow orientation sector), corresponding to a state of minimum entropy. Conversely, the onset of abrupt maneuvers, occlusions, or background clutter causes the motion vectors to scatter across various directions, sharply increasing informational uncertainty and, consequently, the entropy value—signaling a transition to a "chaotic" state. Utilizing entropy as a descriptor allows for the proactive detection of tracking stability loss, often before the tracking algorithm itself fails due to cumulative errors [15].

Motion Entropy Estimation

We now describe how we compute the motion entropy that drives the scheduler. The goal is to produce a scalar H_t at each frame t that quantifies the unpredictability or disorder in the target's recent motion. Our approach is to analyze the orientation distribution of the target's frame-to-frame displacement vectors and compute the Shannon entropy of that distribution [14].

Firstly, we obtain the target's motion vectors over a short time window. In our implementation, we use the Kalman filter's estimated velocities as well as optical flow within the target region to characterize motion. Specifically, between frame $t-1$ and t, we take the Kalman-predicted displacement $(\Delta x, \Delta y)$ (which is also like the lightweight correlation-filter tracker's observed displacement if tracking was successful). Additionally, we compute a sparse optical flow using Shi-Tomasi corners [16] within the target's bounding box – this gives us multiple motion vectors v_i for different parts of the target or local background. The collection of motion vectors is then projected to orientations (angles). We accumulate these angles into a histogram with N bins covering $[0, 2\pi)$. Based on prior studies [17], we choose $N = 16$ orientation bins (each 22.5° wide), which provide a good balance between resolution and statistical reliability. Let n_k be the count (or sum of weights) of motion vectors falling into the bin k. We weigh each vector's contribution by its magnitude, so that larger motions influence the entropy more than tiny jitter (this is like the motion magnitude weighting proposed by Chen et al. [17]). After weighing, we compute the probabilities:

$$p_k = n_k / \sum_{j=1}^{N} n_j. \quad (1)$$

This yields a normalized distribution over orientations.

Shannon Entropy Calculation

The Shannon entropy [14] of the orientation distribution is then:

$$H = -\sum_{j=1}^{N} p_k \log p_k. \quad (2)$$

We further normalize this value by the maximum entropy $\log N$ (achieved when the distribution is uniform) to get a normalized entropy:

$$H_{norm} = H / \log N, \quad (3)$$

so that it lies between 0 and 1. In this normalized scale, $H_{\text{norm}} \sim 0$ means all motion vectors point to the same direction (perfectly regular motion), while $H_{\text{norm}} = 1$ means the motion vectors are uniformly distributed across all directions (completely random or highly complex motion). We found this normalization helpful for defining general thresholds that transfer across scenarios. We maintain a smoothed entropy value and update it recursively:

$$\widetilde{H}_t = \beta \cdot \widetilde{H}_{t-1} + (1 - \beta) \cdot H_{\text{norm},t}, \quad (4)$$

where $\beta \in [0,1)$ is a smoothing factor (we used $\beta = 0.8$ so that the entropy effectively averages over the last few frames). This smoothing dampens transient spikes in entropy and captures the general trend of motion complexity. A sudden one-frame orientation change won't immediately flip the scheduler, unless it persists or continues chaotic. The choice β can be adjusted; a higher β (closer to 1) value means slower adaptation (more inertia in the decision), and a lower β value responds quicker but potentially more oversensitive.

Magnitude Thresholding and Residual Flow

We ignore very small motions when computing entropy. If the target moved less than a few pixels (below a threshold) in a frame, we do not treat that as a meaningful motion direction. In practice, if the displacement magnitude of a vector is below, say, 1 pixel (accounting for noise or static target), we exclude it from the orientation histogram. If all motion vectors in a frame are below this threshold (target basically stationary), we define entropy as $H_{\text{t}} = 0$ by default – there is no unpredictability in not moving. This prevents random orientation values due to noise from artificially inflating entropy when the target is actually static.

In UAV footage, a large portion of the optical flow is caused by the camera itself (translation, yaw, small vibrations), so entropy computed directly from raw vectors may reflect platform motion rather than target dynamics. To isolate the target, we first estimate a dominant global motion field from background feature tracks outside the target box. We then subtract this global component from each motion vector inside the target region and

build the orientation histogram from the remaining residual vectors. In parallel, we compute an entropy value for the global flow field; subtracting this baseline from the target-region entropy yields a relative residual-flow entropy that stays low during smooth flight but rises when local motion becomes inconsistent (abrupt maneuvers, partial occlusion, distractors). The resulting value is clipped to the normalized range and used as a frame-reliability cue (e.g., to avoid updating the low-cost tracker during camera shake).

After these steps, we obtain the smoothed motion entropy H_t for each frame. This value will be used by the scheduler logic. We determine two key hyperparameters related to H: a high threshold T_{high} for switching to the resource-intensive tracker, and a low threshold T_{low} for switching back to the lightweight correlation-filter tracker. Additionally, we set a cooldown interval to avoid switching too frequently (even with hysteresis, we impose that once a switch occurs, the system waits at least e.g., 5 frames before another switch).

Entropy-Guided Scheduler Logic

With the smoothed entropy signal $\widetilde{H}_t$ in hand, the scheduler applies a hysteresis rule to decide the tracker mode. We maintain a binary mode state in [Lightweight tracker, High-accuracy tracker], indicating which tracker is currently in control of the target output. The state transition logic is:

Lightweight correlation-filter tracker mode (normal): Remain in lightweight correlation-filter tracker mode as long as $\widetilde{H}_t < T$. If the condition $\widetilde{H}_t > T$ is sustained for a certain period (we require it to be above for at least 5 consecutive frames to avoid jitter trigger), then transition to deep neural tracker mode. When transitioning, signal the deep neural tracker to start tracking from the current frame t onward. We also optionally take the current KCF state (position, size) and refine it with the deep neural tracker immediately to avoid any one-frame delay. In practice, we run the deep neural tracker on the same frame t as soon as the decision is made, so we get the benefit of the deep tracker without waiting for the next frame. The mode switch triggers a reset of the cooldown timer.

Resource-intensive tracker mode (robust): Remain in resource-intensive tracker mode if $\widetilde{H}_t > T$ the condition is true continuously for a short window (again, we use a 5-frame confirmation), then transition back to lightweight correlation-filter tracker mode. Upon transitioning, we reinitialize or update the lightweight correlation-filter tracker with the resource-intensive tracker's latest bounding box. Specifically, we re-center KCF's filter on the resource-intensive tracker position and if possible, update its appearance model with the current frame's target patch (to catch up any appearance changes that happened during resource-intensive tracker mode). After that, the resource-intensive tracker is put on standby (it may stop running to save computation). The mode switch also resets the cooldown if motion entropy drops below T_{low}.

The hysteresis ensures that once we switch to a deep neural tracker due to high entropy, we don't immediately flip back to a lightweight tracker at the first moment entropy dips, which could be an outlier. Likewise, after switching back to the lightweight tracker, we require a significant entropy rise to switch again. We set $T_{high} > T_{low}$ it so there is a clear gap. In our default, 0.65 vs 0.50 (see **Table 1**), which worked well. The cooldown of 5 frames added a further guard: for example, if entropy oscillates around 0.6, the system might switch to deep neural tracker at 0.65, then entropy dips to 0.6 (still above 0.5, so it stays deep neural tracker), then maybe rises again – the cooldown ensures we don't switch off deep neural tracker too quickly and then on again. Essentially, once a deep neural tracker is engaged, we want to stick with it for a reasonable duration (at least a few frames) to see through the turbulent period.

Table 1. **Entropy-related parameters.**

Parameter	Symbol	Value (default)	Description
Orientation histogram bins	N	16	Number of bins for motion direction (0–360°).
Smoothing factor	β	0.8	Exponential smoothing weight for H_t.
High entropy threshold	T_{high}	0.65	Threshold to trigger the switch to the high-accuracy tracker (dimensionless, between 0 and 1).
Low entropy threshold	T_{low}	0.50	Threshold to trigger the switch back to the lightweight tracker.
Motion ignore threshold	–	1 px/frame	Motions below this pixel magnitude are treated as zero motion (noise filter).
Switch frames threshold	–	5 frames	Minimum number of frames to wait after a switch before another switch is allowed.

While in lightweight correlation-filter tracker mode, the resource-intensive tracker could either be completely off or running at a low frequency. In our implementation, we opted to run the resource-intensive tracker at a very low frequency (e.g., once every 10 frames) even in lightweight correlation-filter tracker mode, just to maintain an updated idea of the target's appearance in case it has changed significantly. This is not strictly necessary; one could turn off the resource-intensive tracker entirely to save power and only initialize it when needed (incurring a small initiation cost). We found the overhead of running it occasionally was minimal, and it provided a slight safety net (if KCF was close to failure, the resource-intensive tracker might already be able to pick up immediately). However, for simplicity, one can imagine the resource-intensive tracker is effectively off during lightweight correlation-filter tracker mode.

Conversely, while in resource-intensive tracker mode, we often still run the lightweight correlation-filter tracker in the background. The lightweight correlation-filter tracker might fail during this period (since presumably entropy was high for a reason), but we let it continue updating with the Kalman predictions and occasional corrections from resource-intensive tracker output. The benefit is that when we switch back to the lightweight correlation-filter tracker, it can resume without a full re-initialization from scratch. In some cases, if the lightweight correlation-filter tracker completely lost the target during resource-intensive tracker mode (e.g., KCF drifted to background because we stopped updating it), we simply reinitialize KCF at switch-back time, which is essentially instantaneous.

Implementation and Platform

We implemented the proposed framework using Python 3.10, and the Siamese resource-intensive tracker was executed with PyTorch 2.1.0. The experiments were conducted in the Google Colab virtual environment. Therefore, the reported results should be interpreted within the context of a simulation-based setup for resource-constrained UAVs, rather than as an execution on actual onboard drone hardware. The setup reflected

a lightweight CPU stage (KCF, Kalman prediction, optical flow) and a GPU-accelerated deep-learning stage, but it did not represent measurements on a specific UAV hardware platform. The allocated session provided hardware resources roughly equivalent to an Intel Xeon-class 2 vCPU CPU, an NVIDIA T4 GPU with 16 GB VRAM, and approximately 12.7 GB of RAM. No explicit instruction-set restrictions were applied. The baseline FPS for each tracker was measured by running it on the entire dataset and averaging the rate. Note that for the hybrid algorithm, the FPS can vary sequence-to-sequence depending on how often the resource-intensive tracker is invoked; we report the overall average and discuss the range. Actual onboard deployment would require additional profiling on the target UAV platform under its memory and power constraints.

We tuned the hyperparameters (entropy thresholds, etc.) empirically and fixed them for all results presented. For APCE-Hybrid, we set the APCE threshold analogously to the entropy-based threshold so that it triggers the resource-intensive tracker under comparable conditions of tracking difficulty. We calibrated this threshold by identifying APCE values that consistently decreased shortly before KCF lost the target.

RESULTS AND DISCUSSION

Datasets and Baselines

We evaluate our approach on two standard single-object tracking datasets that include UAV scenarios and varied difficulties: UAV123 [18] is a UAV-specific benchmark with 123 aerial video sequences (over 110K frames) capturing objects such as cars, boats, and people from a drone perspective. It includes challenging attributes like fast motion, camera motion, small objects, and occlusion – reflecting real UAV tracking conditions; OTB100 [19] is the Object Tracking Benchmark (100 sequences) widely used in tracking literature; it contains various scenes (not UAV-specific) and provides established evaluation protocols (success and precision). These two datasets cover a broad spectrum from short-term to long-term tracking scenarios.

We compare the proposed entropy-hybrid tracker with the following baselines (which are modern tiny state-of-the-art models to run in real-time on resource-constrained devices):

1. KCF – the baseline correlation filter tracker without any switching (essentially our lightweight correlation-filter tracker running alone).
2. MobileTrack – a recent efficient Siamese tracker optimized for UAVs, representing the state-of-the-art in high-speed tracking (we use an implementation of MobileTrack for comparisons).
3. TransT – a modern transformer-based tracker [2], representing top-tier accuracy (albeit at a higher computational cost). TransT does not impose real-time constraints on our platform but gives an upper bound on accuracy for reference.
4. Fixed periodic trigger – a simple hybrid baseline that activates the deep neural tracker at a fixed interval (every t frames) regardless of confidence or entropy, similar to periodic verification strategies used in long-term tracking [13].
5. APCE-Hybrid as an ablative baseline like Cao et al. [9]: this is our implementation of a tracker that switches between KCF and the Siamese deep neural tracker based on the APCE threshold (rather than entropy). APCE-Hybrid uses the same two trackers as our method but triggers the deep neural tracker when KCF's normalized response peak drops below a set threshold (indicating low confidence). This allows us to contrast entropy vs. direct confidence-based switching.

Metrics

We use standard tracking metrics: Success (AUC) and Precision. Success is measured by the Intersection-over-Union (IoU) overlap between the predicted bounding

box and ground truth, and the success rate is the fraction of frames where IoU exceeds a threshold. By varying the threshold from 0 to 1, an area under the curve (AUC) score is computed. We report the AUC as a summary of overall accuracy. Precision is measured by Center Location Error (CLE): the percentage of frames where the predicted center is within a certain distance (typically 20 pixels) of the ground truth center [19]. We report the precision at a 20px threshold, following the OTB100 benchmarking methodology, as well as the mean CLE. Additionally, since our focus is on real-time performance, we report the average Frames Per Second (FPS) each tracker runs at (on a given hardware configuration), and an estimate of computational complexity in FLOPs (floating point operations per frame). The FLOPs give a hardware-independent measure of efficiency: we calculate it by summing the major operations of each tracker per frame (for deep trackers, this includes convolution operations; for KCF, it's negligible). For our hybrid, the effective FLOPs per frame are lower than those of the deep tracker since the deep component runs intermittently.

Accuracy and Speed Comparison

Table 2 reports the overall performance of our Entropy-Guided Hybrid tracker against the baselines on the UAV123 and OTB100 datasets. We list the Success AUC, Precision (20px), FPS, and average GFLOPs per frame for each method.

Table 2. **Performance comparison of the proposed entropy-guided switching method versus baseline trackers on UAV123 and OTB100.**

Tracker	UAV123		OTB100		FPS	FLOPs (G)
	Success AUC	Precision	Success AUC	Precision		
KCF	0.432	51.7%	0.521	63.4%	160	~0.02
MobileTrack (2022)	0.690	77.3%	0.617	81.1%	80	~1.2
TransT (2021)	0.717	81.4%	0.754	85.0%	35	~8.0
Fixed periodic trigger (2019)	0.555	65.0%	0.582	67.5%	95	~0.8
APCE Hybrid (2025)	0.573	66.5%	0.590	69.5%	110	~0.9
Entropy-Hybrid (Ours)	0.594	69.5%	0.601	72.0%	100	~0.6

Our entropy-hybrid tracker improves substantially over KCF on UAV123, increasing Success AUC from 0.432 to 0.594 and Precision from 51.7% to 69.5%. Although always-on deep trackers still lead in pure accuracy (MobileTrack: 0.690 AUC, 77.3% precision; TransT: 0.717 AUC, 81.4% precision), our hybrid achieves a better efficiency–accuracy balance, running at 100 FPS with ~0.6 GFLOPs per frame versus 80 FPS / ~1.2 GFLOPs for MobileTrack and 35 FPS / ~8.0 GFLOPs for TransT.

Compared to the fixed periodic trigger baseline [13], entropy-based switching yields higher accuracy (0.594 vs 0.555 AUC on UAV123) while also lowering average compute (~0.6 vs ~0.8 GFLOPs) by avoiding unnecessary deep activations during stable motion.

Against APCE-Hybrid, our method gains 0.021 AUC and 3.0 percentage points precision on UAV123 (0.573→0.594, 66.5%→69.5%) with a notable reduction in average computation (~0.9→~0.6 GFLOPs). This supports the idea that motion entropy can trigger the resource-intensive tracker more proactively than response-peak degradation, improving continuity before the low-cost tracker drifts.

On OTB100, the entropy hybrid is closed in accuracy to MobileTrack in AUC (0.617 vs 0.601) while keeping a higher frame rate (100 vs 80 FPS). The relative gain over KCF remains clear (0.521→0.601 AUC), indicating that switching is beneficial beyond UAV-specific footage, even though OTB sequences often contain longer low-entropy intervals where the system stays in lightweight mode.

Ablation Analysis

We perform an ablation study to isolate the impact of our entropy-guided switching. We compare four configurations: (A) KCF-only, (B) deep neural tracker-only (MobileTrack every frame), (C) APCE-triggered hybrid, and (D) Entropy-triggered hybrid (ours). Configuration A and B represent the extremes of never switching (always using one tracker). C and D use the same components but different switching signals.

Table 3 confirms that the switching mechanism is crucial. Either hybrid vastly outperforms the single trackers, proving the effectiveness of combining trackers. Between triggers, the entropy-based scheduler provides a better balance, improving accuracy the most with only minimal overhead. It validates our choice of using motion entropy as a superior switching signal compared to solely relying on the tracker's internal confidence.

Table 3. **Ablation of tracker switching strategies on UAV123.**

Strategy	AUC	Precision	Avg FPS	Notes
A. KCF-only (no switch)	0.432	51.7%	160	Fast but loses target often.
B. Deep neural tracker-only (Siamese every frame)	0.690	77.3%	80	Accurate, but always computationally intensive.
C. Hybrid (APCE switch)	0.573	66.5%	110	Switches on KCF failure (reactive).
D. Hybrid (Entropy switch)	0.594	69.5%	100	Switches on motion entropy (proactive).

Discussion

The proposed approach demonstrates that an information-theoretic measure like entropy can be highly useful in a control loop for visual tracking. By focusing on motion characteristics (which are agnostic to appearance), our scheduler gains a kind of foresight into tracking difficulty. This is particularly beneficial in UAV scenarios where camera motion and target motion interplay. For instance, if a drone suddenly turns, causing the background to shift in a new direction relative to the target, the entropy of the optical flow in the scene spikes – our system notices this and preemptively bolsters the tracking with the Siamese network. This behavior is a distinct advantage over reactive schemes that wait for the primary tracker to fail. In effect, motion entropy serves as a proxy for "how hard is the target to follow right now," encapsulating factors like erratic

target maneuvering or complex background motions. This work suggests that integrating such high-level cues can make traditional trackers smarter without explicitly training a switching classifier or network.

Our results demonstrate the trade-off between efficiency and accuracy in edge deployment. Running a top-tier tracker (TransT) would give the best accuracy, but at an untenable speed/power cost on UAV hardware. Running a lightweight correlation-filter tracker (KCF) yields great speed but will miss the target often. By spending computation only on the hard frames, we allocate resources dynamically where they have the most impact, achieving an overall efficiency that would be impossible if the resource-intensive tracker ran uniformly. This is particularly important for UAVs running on battery – computational load often directly translates to battery drain and flight time limitations. Our method allows the system to cruise on low power most of the time and only spike usage occasionally.

While our Kalman filter integration improves short gaps handling, it does introduce a risk: if the target is fully occluded and the Kalman filter propagates the state without measurement updates for too long, it can drift off course. In our tests, if the occlusion lasted beyond about 1 second, the Kalman prediction became unreliable. If the target reappears far from the predicted spot, KCF might not search far enough to find it. Our current system mitigates this by relying on the high-accuracy tracker in such cases – the entropy typically would be high during the occlusion onset (e.g., when the target disappears behind an object, there is often a flurry of motion edges), triggering the high-accuracy tracker. The high-accuracy tracker's larger search window and template matching can reacquire the target even if it breaks linear motion assumptions. However, in extreme cases where the target stops moving entirely behind an occluder (zero motion, so low entropy, and Kalman just coasts), neither tracker may see it until it reappears. If the reappearance is significantly off the predicted path, our current strategy might not catch it immediately. This is where integrating a dedicated re-detector (like a YOLO) could help. In future extensions, we could incorporate a third component: if the target is not found by either tracker for a certain period, trigger a full-frame detection. This would handle total failures. The trade-off is the complexity and potential false positives from detection, so we avoided it in this work.

Introducing entropy shows it works well in our UAV context because it naturally focuses on residual motion rather than pure camera-induced motion. When the entire scene moves uniformly (due to drone translation), the optical flow vectors are largely aligned and have low entropy. In our implementation, we first approximate a dominant background flow vector in the region and subtract it, computing entropy on the residual flow field. This step suppresses global jitter and small handshake effects from the UAV device. But if the motion is globally consistent, KCF (augmented with a Kalman filter applied for motion prediction) can handle it well. It is the irregular residual motion that causes trouble (e.g., the target abruptly changing direction or multiple objects crossing paths and generating flows in different directions), and the entropy metric cleanly reflects that irregularity.

CONCLUSION

We presented an entropy-guided hybrid tracker for real-time UAV target tracking that dynamically switches between a fast correlation filter tracker and a deep Siamese tracker. The central contribution is the introduction of motion entropy as a decision signal to anticipate tracking difficulty. By measuring the randomness of recent target motion, our scheduler intelligently allocates computational effort: it keeps the lightweight correlation-filter tracker in charge during easy, predictable segments. It activates the resource-intensive tracker during challenging maneuvers or occlusions. This approach

tackles the edge computing paradox by achieving high accuracy when needed without continuously running expensive algorithms.

Through experiments on several benchmarks, we demonstrated that our method delivers a balance of accuracy and efficiency. On UAV123, it improved tracking success by 0.162 AUC compared to a baseline KCF (0.432→0.594) while maintaining ~100 FPS. The hybrid also outperformed both a fixed periodic trigger baseline and a purely confidence-based switching baseline, highlighting the efficacy of the entropy cue. These results indicate that motion entropy is a practical heuristic for tracker management that adapts its strategy based on observed dynamics.

In practical terms, our system can extend the deployment of advanced tracking on resource-limited UAV platforms. A drone using this tracker can handle fast-moving or erratic targets with accuracy nearly on par with modern deep trackers, yet for much of the flight, it only expends the computation of a tiny fraction of a deep network. This has direct implications for onboard energy consumption and responsiveness.

There are several future directions for this work. One is to integrate an object detector as a fail-safe for prolonged target loss, combining detection with our tracking switcher to handle complete occlusions or exits from the field of view. Another direction is to make the entropy scheduler a neural network which can be trained to better determine complex scenes and further apply self-learning algorithms to improve the model on fly.

In conclusion, entropy-guided tracker switching offers a promising way to satisfy the competing demands of accuracy and efficiency in UAV target tracking. It uses a broader principle of adaptive edge AI: using inexpensive computations to decide when to deploy expensive ones. We believe this paradigm can be applied to many complex problems on resource-constrained platforms in real-time, enabling smarter and more autonomous systems without hardware upgrades.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, writing, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [V.O.]; methodology, [V.O.]; validation, [V.O.]; writing – original draft preparation, [V.O.]; writing – review and editing, [S.V., V.O.]; supervision, [S.V.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Singh, A., Saini, K., Nagar, V., Aseri, V., Sankhla, M. S., Pandit, P. P., & Chopade, R. L. (2022). Artificial intelligence in edge devices. Advances in Computers, 127, 437–484. https://doi.org/10.1016/bs.adcom.2022.02.013

[2] Chen, X., Yan, B., Zhu, J., Wang, D., Wang, X., & Lu, H. (2021). Transformer tracking. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 8126–8135. https://doi.org/10.1109/CVPR46437.2021.00803

[3] Xue, Y., Jin, G., Shen, T., Tan, L., Yang, J., & Hou, X. (2022). MobileTrack: Siamese efficient single object tracker for high-speed UAV tracking. IET Image Processing, 16(12), 3300–3313. https://doi.org/10.1049/ipr2.12565

[4] Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2015). High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(3), 583–596. https://doi.org/10.1109/TPAMI.2014.2345390
[5] Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2544–2550. https://doi.org/10.1109/CVPR.2010.5539960
[6] Wu, P., Li, Y., & Xue, D. (2025). UAV target tracking: A survey. Artificial Intelligence Review, 58(11), 1–41. https://doi.org/10.1007/s10462-025-11348-x
[7] Zhang, Y., Yang, Y., Zhou, W., Shi, L., & Li, D. (2018). Motion-Aware Correlation Filters for Online Visual Tracking. Sensors, 18(11), 3937. https://doi.org/10.3390/s18113937
[8] Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7), 1409–1422. https://doi.org/10.1109/TPAMI.2011.239
[9] Cao, S., et al. (2025). UAV real-time target detection and tracking algorithm based on improved KCF and YOLOv5s_MSES. Machines, 13(5), 364. https://doi.org/10.3390/machines13050364
[10] Ai, Y., et al. (2025). Real-time occluded target cooperative tracking method for UAVs. Electronics, 14(20), 4034. https://doi.org/10.3390/electronics14204034
[11] Wang, J., Liu, Y., Ai, Y., & Xue, W. (2021). Long-term target tracking combined with re-detection. EURASIP Journal on Advances in Signal Processing, 2021, 79. https://doi.org/10.1186/s13634-020-00713-3
[12] Ma, H., Acton, S. T., & Lin, Z. (2020). SITUP: Scale invariant tracking using average peak-to-correlation energy. IEEE Transactions on Image Processing, 29, 3546–3557. https://doi.org/10.1109/TIP.2019.2962694
[13] Liu, F., Mao, K., Qi, H., & Liu, S. (2019). Real-time long-term correlation tracking by single-shot multibox detection. Optical Engineering, 58(1), 013105. https://doi.org/10.1117/1.OE.58.1.013105
[14] Shannon, C. E. (1948). A mathematical theory of communication. Bell System Technical Journal, 27(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
[15] Jaynes, E. T. (1957). Information theory and statistical mechanics. Physical Review, 106(4), 620–630. https://doi.org/10.1103/PhysRev.106.620
[16] Shi, J., & Tomasi, C. (1994). Good features to track. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 593–600. https://doi.org/10.1109/CVPR.1994.323794
[17] Chen, C.-Y., et al. (2008). Motion entropy feature and its applications to event-based segmentation of sports video. EURASIP Journal on Image and Video Processing, 2008, 460913. https://doi.org/10.1155/2008/460913
[18] Mueller, M., Smith, N., & Ghanem, B. (2016). A benchmark and simulator for UAV tracking. European Conference on Computer Vision (ECCV), 445–461. https://doi.org/10.1007/978-3-319-46448-0_27
[19] Wu, Y., Lim, J., & Yang, M.-H. (2015). Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9), 1834–1848. https://doi.org/10.1109/TPAMI.2014.2388226
[20] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. (2016). Fully-convolutional siamese networks for object tracking. European Conference on

Computer Vision Workshops (ECCVW), 850–865. https://doi.org/10.1007/978-3-319-48881-3_56
[21] Ye, B., Chang, H., Ma, B., Shan, S., & Chen, X. (2022). Joint feature learning and relation modeling for tracking: A one-stream framework. Computer Vision – ECCV 2022, 341–357. https://doi.org/10.1007/978-3-031-20047-2_20
[22] Cui, Y., Jiang, C., Wu, G., & Wang, L. (2024). MixFormer: End-to-end tracking with iterative mixed attention. IEEE Transactions on Pattern Analysis and Machine Intelligence. https://doi.org/10.1109/TPAMI.2024.3349519
[23] Javed, S., Danelljan, M., Khan, F. S., Khan, M. H., Felsberg, M., & Matas, J. (2023). Visual Object Tracking With Discriminative Filters and Siamese Networks: A Survey and Outlook. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(5), 6552–6574. https://doi.org/10.1109/TPAMI.2022.3212594

КЕРУВАННЯ ПЕРЕМИКАННЯМ ЗАСОБІВ СУПРОВОДУ НА ОСНОВІ ЕНТРОПІЙНОГО АНАЛІЗУ ДЛЯ ВІДСТЕЖЕННЯ ЦІЛЕЙ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ У РЕАЛЬНОМУ ЧАСІ

Володимир Олексюк**, Сергій Вельгош
Кафедра радіофізики та комп'ютерних технологій,
Львівський національний університет імені Івана Франка,
вул. Генерала Тарнавського, 107, 79017 Львів, Україна

АНОТАЦІЯ

Вступ. Автоматичне наведення безпілотних літальних апаратів потребує надійного супроводу цілі в реальному часі на бортовому обладнанні з обмеженими обчислювальними ресурсами. Сучасні засоби супроводу на основі згорткових нейронних мереж і трансформерів забезпечують високу точність, однак для безперервної роботи на периферійних пристроях часто є надто повільними та обчислювально затратними. Натомість методи на основі кореляційних фільтрів працюють із високою частотою кадрів, але можуть накопичувати похибку або втрачати ціль за умов перекриття чи різких маневрів. Цей компроміс між точністю та ресурсоефективністю (парадокс застосування штучного інтелекту на периферійних пристроях) зумовлює потребу в адаптивних підходах, що узгоджують швидкодію та витрати ресурсів і зберігають обчислювальний резерв для інших бортових задач.

Матеріали та методи. Запропоновано метод перемикання засобів супроводу на основі аналізу ентропії, що поєднує швидкий алгоритм на основі кореляційних фільтрів (KCF), доповнений прогнозуванням руху за фільтром Калмана, та точніший глибинний сіамський засіб супроводу. Планувальник ентропії руху оцінює непередбачуваність переміщення цілі за нормованою ентропією Шеннона змін орієнтації руху на останніх кадрах. Для зменшення впливу короткочасних сплесків застосовано експоненційне згладжування, а порогові правила визначають моменти, коли достатньо KCF і коли слід активувати глибинний модуль.

Результати. Під час випробувань на еталонних наборах даних UAV123 і OTB100 гібридний підхід підвищив success AUC приблизно на 10% порівняно з KCF і приблизно 70% від показника трансформерної моделі, при цьому працюючи у 1,5–3 рази швидше, ніж режим із постійно ввімкненим глибинним модулем модуль вмикається лише у складні проміжки, забезпечуючи близько 100 кадрів/с та середні витрати ≈0,6 GFLOPs на кадр проти ≈1–4 GFLOPs для суто глибинних підходів.

Висновки. Розроблений метод відстеження цілі на основі аналізу ентропії руху забезпечує адаптивний компроміс між ефективністю, швидкодією та точністю, підтримуючи високу точність під час маневрів і перекриттів цілі та зменшуючи навантаження за стабільного руху. Запропонований підхід є практичним рішенням для високоефективного супроводу цілей БПЛА на бортових пристроях.

Ключові слова: відстеження цілі; кореляційні фільтри; сіамська нейронна мережа; ентропія руху; гібридний алгоритм супроводу; периферійні обчислення.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
23 February, 2026	20 March, 2026	23 March, 2026	30 March, 2026