remote sensing

MDPI

Article

TPC-Tracker: A Tracker-Predictor Correlation Framework for Latency Compensation in Aerial Tracking

Xuqi Yang [1,2], Yulong Xu [2], Renwu Sun [2], Tong Wang [1] and Ning Zhang [1,*]

[1] Information and Electronics, Beijing Institute of Technology, Beijing 100081, China; xuqiyang@bit.edu.cn (X.Y.); 3120235586@bit.edu.cn (T.W.)
[2] The National Key Laboratory of Intelligent Collaborative Perception and Analytic Cognition, Beijing 100124, China; whnjbj@163.com (Y.X.); sunshu105@126.com (R.S.)
* Correspondence: nzhang.rs@bit.edu.cn

Highlights

What are the main findings?

- Proposes TPC-Tracker, a Tracker-Predictor Correlation Framework specialized for latency compensation in remote sensing-based aerial tracking. It fuses continuous historical visual features from remote sensing imagery with continuous historical motion features, addressing the weak correlation between trackers and predictors in existing methods, addressing the weak correlation between trackers and predictors in existing methods, and reduces the Mean Squared Error (MSE) by up to 38.95% in remote sensing-oriented physical tracking simulations.

What are the implication of the main findings?

- The significant reduction in mean square error of physical tracking confirms that the fusion of continuous historical visual and motion features can effectively alleviate the negative impact of UAV attitude changes and long-term target occlusion on tracking accuracy, laying a foundation for the practical application of high-precision aerial tracking in remote sensing scenarios such as precision agriculture and emergency search and rescue.

Abstract

Online visual object tracking is a critical component of remote sensing-based aerial vehicle physical tracking, enabling applications such as environmental monitoring, target surveillance, and disaster response. In real-world remote sensing scenarios, the inherent processing delay of tracking algorithms results in the tracker's output lagging behind the actual state of the observed scene. This latency not only degrades the accuracy of visual tracking in dynamic remote sensing environments but also impairs the reliability of UAV physical tracking control systems. Although predictive trackers have shown promise in mitigating latency impacts by forecasting target future states, existing methods face two key challenges in remote sensing applications: weak correlation between trackers and predictors, where predictions rely solely on motion information without leveraging rich remote sensing visual features; and inadequate modeling of continuous historical memory from discrete remote sensing data, limiting adaptability to complex spatiotemporal changes. To address these issues, we propose TPC-Tracker, a Tracker-Predictor Correlation Framework tailored for latency compensation in remote sensing-based aerial tracking. A Visual Motion Decoder (VMD) is designed to fuse high-dimensional visual features from remote sensing imagery with motion information, strengthening the tracker-predictor connection. Additionally, the Visual Memory Module (VMM) and Motion Memory Module (M3) model discrete historical remote sensing data into continuous spatiotemporal memory, enhancing

Check for updates

Academic Editor: Pedro Melo-Pinto

Received: 17 December 2025
Revised: 12 January 2026
Accepted: 12 January 2026
Published: 19 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

predictive robustness. Compared with state-of-the-art predictive trackers, TPC-Tracker reduces the Mean Squared Error (MSE) by up to 38.95% in remote sensing-oriented physical tracking simulations. Deployed on VTOL drones, it achieves stable tracking of remote sensing targets at 80 m altitude and 20 m/s speed. Extensive experiments on public UAV remote sensing datasets and real-world remote sensing tasks validate the framework's superiority in handling latency-induced challenges in aerial remote sensing scenarios.

Keywords: predictive tracking; aerial tracking; latency compensation

1. Introduction

Visual object tracking (VOT), which estimates the spatiotemporal state of targets in remote sensing video sequences given initial annotations, is indispensable for UAV-based remote sensing applications such as precision agriculture, urban mapping, and emergency search [1–4]. Recent work on VOT has made significant progress in terms of accuracy and inference speed for normal scenarios through siamese networks [5,6] and transformer techniques [7]. However, in practical remote sensing missions, the time interval between frame capture and algorithm output leads to inevitable latency: by the time tracking results are generated, the remote sensing target and surrounding environment have undergone spatiotemporal changes. This latency will hurt both visual tracking and physical tracking control in remote sensing contexts. Particularly in control-critical remote sensing tasks like real-time environmental monitoring, the mismatch between delayed tracking results and actual remote sensing target states causes substantial control errors [8]. This delay creates a mismatch between predicted and actual object states. In sensor-fusion strategies combining camera data with IMU, Kalman filters receive outdated visual inputs, resulting in inaccurate motion predictions as shown in the Figure 1, while latency-induced errors in object position refinement accumulate in dynamic remote sensing scenes, ultimately resulting in tracking failure.

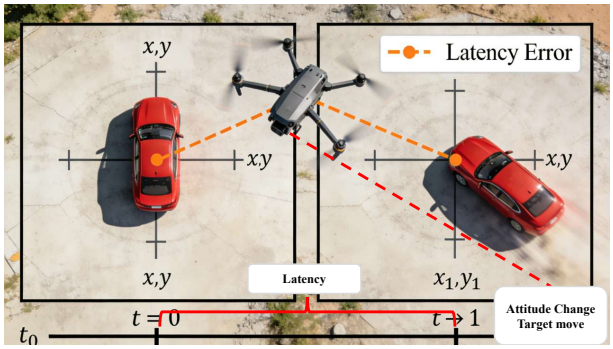


Figure 1. This figure illustrates the latency-induced error in UAV target tracking: At the initial moment (t = 0), the UAV captures an image of the target (red vehicle) and performs perception inference. Due to the latency in sensor data acquisition and algorithmic inference, by the time the inference result is output (t→1), the UAV's own attitude has changed and the target vehicle has moved. This leads to a significant deviation between the target's actual position (x_1,y_1) in the image and the position (x,y) obtained from the inference (marked as "Latency Error" by the orange dashed line). This deviation is the latency error that needs to be addressed.

Two primary research directions address latency in control-sensitive remote sensing applications. The first [9,10] accelerates inference via lightweight model design and deployment, reducing model latency to minimize remote sensing tracking delays. While intuitive, this approach cannot eliminate inherent latency between tracking results and real-world

remote sensing target states, making it insufficient for time-critical remote sensing missions like disaster response and real-time surveillance. The second method [11–15] predicts the future state to shorten the gap between the real state and the tracking result. Ref. [11] gives a latency-aware evaluation to judge the performance, as shown in Figure 2. Although it has extra components to predict, it has more potential because prediction has the potential to eliminate the impact of latency.

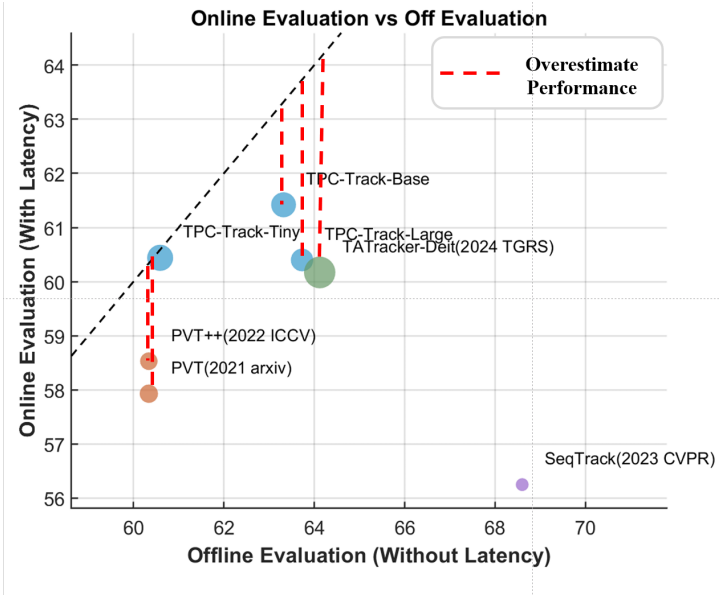


Figure 2. The performance of various trackers in "off-line" and "latency-aware" evaluations, as well as the effect of TPC-Tracker in the public dataset. The bubble size represents the inference speed, where a larger bubble indicates a faster inference speed. The distance from the bubble to the dashed line represents the error caused by online tracking latency compared to offline tracking.

Although predictive tracking technology has broad prospects, there is still much room for improvement in this field for control-sensitive tasks. Existing methods [11,13] merely append a Kalman filter-based predictor to a conventional tracker, where the tracker only needs to input the bounding box into the predictor to yield the prediction result. While easy to implement, this method has notable limitations, its prediction is only applicable to the target motion trajectory under the scenario of fixed camera perspective. In UAV remote sensing missions, however, the platform perspective is in a state of continuous dynamic variation. Such a design that relies solely on bounding boxes and features a complete separation between the tracker and the predictor often leads to low-precision prediction results. To address this issue, refs. [12,14] proposes an improved scheme, enabling the tracker to feed both the target appearance features and the bounding box into the predictor simultaneously, and improving the prediction accuracy through the fusion of these two types of features. Nevertheless, this method still has drawbacks, the input appearance features are both limited to discrete data within a fixed past time window. When the UAV undergoes drastic instantaneous attitude changes or the target suffers from long-term occlusion, these limited discrete historical visual features tend to be extracted as background information. Based on the above analysis, we argue that the predictor should acquire continuous historical visual features that can reflect the target's visual feature since its first appearance from the tracker. Meanwhile, we notice that, similar to the defect of discrete historical visual features, relying solely on bounding boxes within a fixed time window also fails to fully characterize the target motion patterns. It is thus evident that continuous historical motion features of bounding boxes are also crucial. *In summary, this paper aims to construct an efficient system architecture, which leverages the dual drive of*

global continuous historical visual features and global historical motion features to strengthen the feature interaction between the tracker and the predictor, thereby improving the overall tracking and prediction performance.

To address these challenges, we propose TPC-Tracker, a Tracker-Predictor Correlation Framework for latency compensation in remote sensing-based aerial tracking. Firstly, to strengthen the correlation between the tracker and the predictor for remote sensing tasks, the VMD fuses the visual feature from the tracker, the motion memory from the predictor, and the latency to predict the latency compensation. The tracking result is then added to the predicted offset to get the prediction result. Secondly, the Visual Memory Module (VMM) and Motion Memory Module (M3) model discrete historical remote sensing visual and motion information into continuous spatiotemporal memory, enhancing the representational power of remote sensing-derived features. Compared with current predictive trackers, TPC-Tracker has achieved a reduction of up to 38.95% in Mean Squared Error (MSE) during remote sensing-oriented physical tracking simulations. Besides, TPC-Tracker significantly improves the *AUC* 4.9% on UAV123 remote sensing dataset [16] under the Latency-Aware Evaluation [11] in the same device. Results on 3 challenging datasets [8,16,17] demonstrate the superiority of our method in terms of Latency-Aware Evaluation. In real-world remote sensing experiments, TPC-Tracker has been deployed on VTOL drones, achieving stable tracking at an altitude of 80 m and a speed of 20 m/s. Our contributions are fourfold:

(1) A remote sensing-oriented Tracker-Predictor Correlation Framework (TPC-Tracker) is proposed, which strengthens the intrinsic connection between trackers and predictors to significantly improve latency compensation accuracy in aerial remote sensing.
(2) The Visual Memory Module (VMM) and Motion Memory Module (M3) are designed to mine deep spatiotemporal memory from remote sensing data, enhancing predictive performance for dynamic remote sensing targets.
(3) The Visual-Motion Decoder (VMD) fuses remote sensing visual and motion features to strengthen tracker-predictor correlation and generate precise latency compensation offsets.
(4) Comprehensive validation is conducted via physical tracking simulations, latency-aware evaluations on three remote sensing datasets, and real-world deployment on VTOL drones, demonstrating TPC-Tracker's effectiveness in practical remote sensing scenarios.

2. Related Works

To mitigate latency impacts in remote sensing-based aerial tracking, we propose the predictive TPC-Tracker. This section reviews related work from three remote sensing-focused perspectives: (1) Visual Tracking and Physical Tracking; (2) Predictive Tracking; and (3) Temporal Information Exploitation.

2.1. Visual Tracking and Physical Tracking

In object tracking, visual tracking, and analysis of captured camera data as the foundation. Traditional methods like MOSSE [18] and KCF [19] face issues with object deformation, occlusion, and high computational costs. Deep-learning-based approaches, such as Siamese networks [20–23] and Transformer-based models [10,24], enhance generalization but often struggle with real-time performance. To accelerate the inference speed, many efficient architectures are designed for tracking. The Spiking Neural Network, as the third generation of artificial neural networks, is designed with significant inspiration from biological neural networks. The application [25–28] in remote sensing has proven its effectiveness. However, the latency between visual tracking results and real-world scenarios critically impacts subsequent physical tracking.

As demonstrated in [29,30], this delay can cause significant errors in physical tracking algorithms, including sensor-fusion methods using IMU or radar with Kalman filters, and

model-based physical tracking. Outdated visual inputs lead to inaccurate motion predictions and position refinement failures, potentially causing complete tracking breakdowns.

Thus, reducing visual tracking atency is critical for reliable physical tracking in remote sensing applications. Minimizing the detection-output time gap enables effective utilization of remote sensing sensor data and physical models. While inference acceleration mitigates latency, hardware constraints and algorithmic limits impose fundamental bounds on real-time responsiveness for remote sensing tasks. Predicting future target states emerges as an essential approach to transcend this barrier in remote sensing contexts.

2.2. Predictive Tracking

Integrating a predictor to enhance precision [14,15,31–33] represents a well-established technical approach for the latency challenge. Although the methods of predictive tracking are evolving, there has never been a quantitative evaluation standard. To quantify the effect of latency awareness, PVT [11] first proposed the latency-aware evaluation as the benchmark. To make it more in line with the actual situation, PVT++ [12] proposed the Extended Latency-aware Evaluation. In addition, the PVT++ [12] predicts the motion which is the offset between the current frame and the future frames, by learning historical motion and historical visual features in the past k frames.

Despite these advances, existing methods simply concatenate trackers and predictors for remote sensing target state prediction, ignoring their intrinsic correlation. Remote sensing trackers extract appearance features specific to aerial imagery, while predictors focus on motion patterns of remote sensing targets. In contrast, our method mutually fuses remote sensing-derived visual features from trackers and motion features from predictors to more accurately predict target motion states in dynamic remote sensing environments.

2.3. Temporal Information Exploitation

Temporal information mining is pivotal in object tracking, as it unlocks the sequential dependencies within video data. Prior works, such as STARK [34] with its memory-based feature retrieval and TCTracker [35] leveraging Transformer's self-attention, have improved tracking under occlusion and complex motions. However, these methods often treat video frames as discrete units, failing to model continuous motion dynamics. This limitation hinders the accurate prediction of rapidly moving or abruptly changing objects.

Our TPC-Tracker addresses these limitations with two remote sensing-optimized strategies. First, it employs a continuous temporal query mechanism that structures historical remote sensing information to adaptively update target states and mitigate latency. Second, it integrates joint visual-motion temporal modeling, capturing both appearance-based visual features like remote sensing texture and motion-related physical characteristics like target velocity of remote sensing targets. This dual-stream approach enables precise future state predictions, bridging the gap between discrete frame analysis and continuous remote sensing scene dynamics.

3. Preliminary: Latency-Aware Tracker

This section focuses on two key elements. It first presents a method to replicate real-world latency on datasets by Latency-aware Evaluation, following the framework in [11] as shown in Figure 3.

Second, this section defines essential notations within the latency-aware tracking framework, providing a standardized basis for subsequent discussions and formulations.

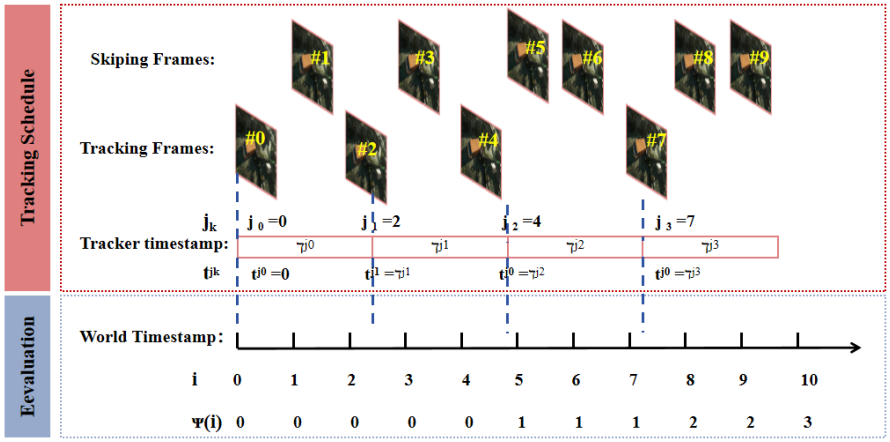


Figure 3. Here is an example of the Latency-aware Evaluation (LAE). When the tracker's speed is slower than the world frame rate, some frames will be skipped. Additionally, there is a discrepancy between the evaluated frame i and the latest tracked frame $\psi(i)$.

In the offline-tracking benchmark, when the frames per second (FPS) acquired by the camera is κ, the object ground-truth g_i and tracking results p_i of each frame can be defined as

$$g_i = [i, x_g^i, y_g^i, w_g^i, h_g^i, t_g^i], \tag{1}$$

$$G = [g_0, g_1, \ldots g_{T-1}], \tag{2}$$

$$p_i = [i, x_p^i, y_p^i, w_p^i, h_p^i, t_p^i], \tag{3}$$

$$P = [p_0, p_1, \ldots p_{T-1}], \tag{4}$$

where i and T denote frame number and sequence length, $x_g^i, y_g^i, w_g^i, h_g^i$ indicates the ground-truth bounding box in i-th frame, and $t_g^i = \frac{i}{\kappa}$ is the worldstamp. The G and P are the set of object ground-truth g_i and tracking results p_i.

In real-world tracking, a slow tracker with significant latency can simply handle the latest frame and discard the frames that are captured during the computation process. The object ground truth of each frame can be defined as

$$r_k = [j_k, x_r^{jk}, y_r^{jk}, w_r^{jk}, h_r^{jk}, t_r^{jk}], \tag{5}$$

$$R = [r_0, r_1, \ldots r_{T-1}], \tag{6}$$

where k denotes the k-th processed frames with a length of K and j_k denotes the world frame number. $x_r^{jk}, y_r^{jk}, w_r^{jk}, h_r^{jk}$ are the result of tracker head on frame j_k. t_r^{jk} is the tracker timestamp, which is the time after predicting the frame j_k. In addition, τ^{jk} is defined as the inference time of frame j_k. As the result, t_r^{jk} can be represented as

$$t_r^{jk} = \sum_{x=0}^{k} \tau^{jk}. \tag{7}$$

When the frame x is skipped, the r_{jk} doesn't exist. The LAE pairs ground-truth $x_g^i, y_g^i, w_g^i, h_g^i$ at time t_g^i with the latest output result $x_r^{\psi(i)}, y_r^{\psi(i)}, w_r^{\psi(i)}, h_r^{\psi(i)}$ where the $\psi(i)$ is

$$\psi(i) = \begin{cases} 0 & , t_r^{jk} \geq t_g^i \\ arg\max_{j_k} t_r^{jk} \leq t_g^i & , \text{others} \end{cases} \tag{8}$$

As a result, the prediction result for LAE can be represented as

$$LAE(p_i) = [i, x_p^{\psi(i)}, y_p^{\psi(i)}, w_p^{\psi(i)}, h_p^{\psi(i)}, t_p^{\psi(i)}], \tag{9}$$

$$LAE(P) = [p_0, p_1, \ldots p_{T-1}]. \tag{10}$$

where i and T denote frame number and sequence length.

4. TPC-Tracker

In this section, a detailed explanation of the proposed TPC-Tracker is provided. First, the overall framework is introduced in Section 4.1. Next, the Visual Memory Module (VMM) in Tracker is described in Section 4.2. Then, the Motion Memory Module (M3) is introduced in Section 4.3. Finally, the Visual Motion Decoder in the predictor is introduced in Section 4.4.

4.1. Overall Framework

As shown in Figure 4, TPC-Tracker is a predictive Tracker, which contains three key modules: the Visual Memory Module (VMM), the Motion Memory Module (M3), and the Visual Motion Decoder (VMD).

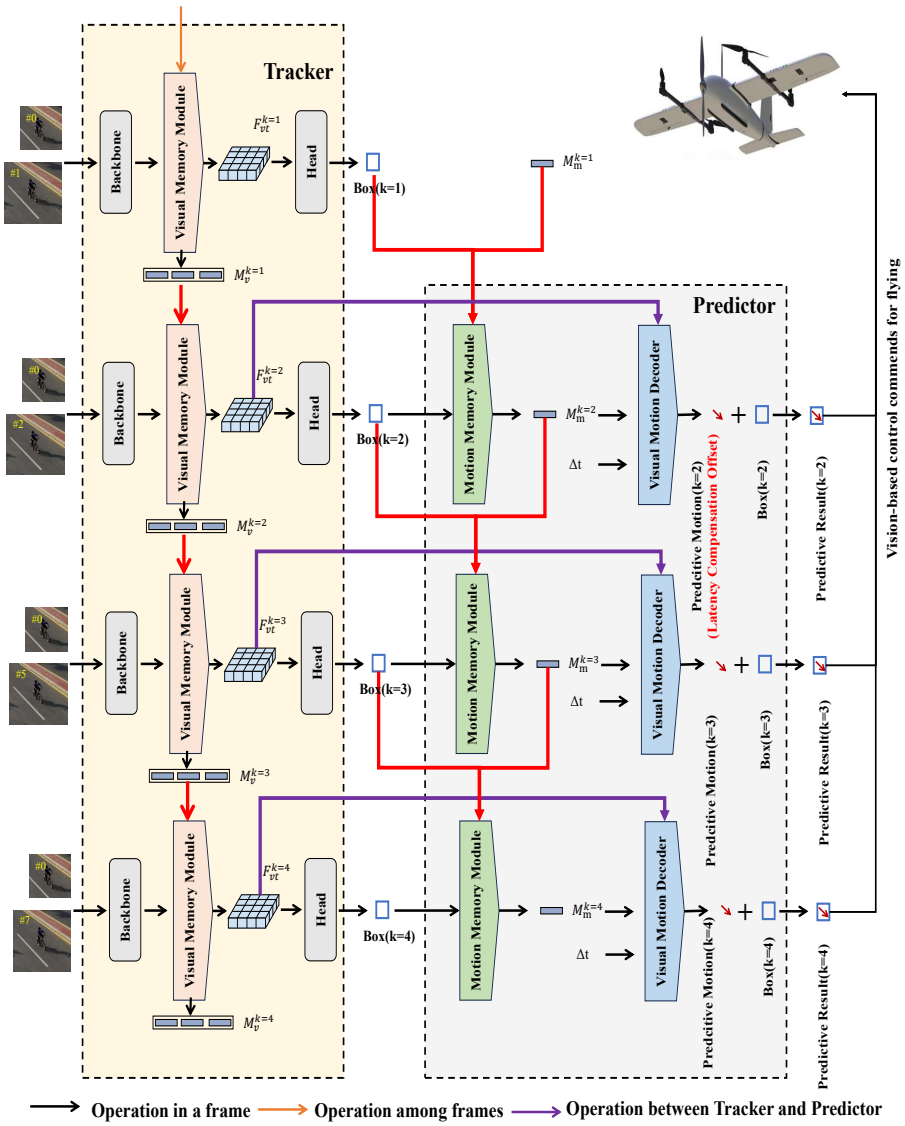


Figure 4. Overview of our TPC-Tracker framework.

4.1.1. Tracker

In the tracker T, the image patch I_{ZX} (including the search region I_X and the template I_Z) is fed into the backbone to get the feature pyramid $F_v = [F_x^{s1}, F_x^{s2}, F_x^{s3}]$. And then, in the i-th frame, the visual feature and the last visual memory $\mathbf{M}_v^i$ are input into the VMM to generate the visual temporal feature F_{vt}^i and update the current visual memory $\mathbf{M}_v^i$. For the two functions, it can be represented as

$$\mathbf{F}_{vt}^i = \varpi_{VMM}^V(F_v, \mathbf{M}_v^{i-1}). \tag{11}$$

$$\mathbf{M}_v^i = \varpi_{VMM}^M(F_v, \mathbf{M}_v^{i-1}). \tag{12}$$

where the $\varpi_{VMM}^V(.)$ represents VMM to generate the visual temporal feature and $\varpi_{VMM}^M(.)$ represents VMM to update the current visual memory.

Then the head H gives the tracking boxes B by classifying and regressing the visual temporal feature as follows:

$$B_i = H(\mathbf{F}_{vt}^i) \tag{13}$$

4.1.2. Predictor

In the predictor P, the current tracking result B_i, the last result B_{i-1}, and the last motion memory M_m^{i-1} are fed into the M3 to update the current motion memory M_m^i. The process can be represented as

$$\mathbf{M}_m^i = \mu_{M3}(B_i, B_{i-1}, \mathbf{M}_m^{i-1}) \tag{14}$$

where $\mu_{M3}(.)$ represents the mapping function of M3.

In addition, the latency Δt, the current motion memory $\mathbf{M}_m^i$, and the visual temporal feature $\mathbf{F}_{vt}^i$ are put into VMD to fuse the visual information and motion information and predict the motion offset Motion $\mathcal{M}_i$ during the latency. The process can be represented as

$$\mathcal{M}_i = \wp_{VMD}(\Delta t, \mathbf{F}_{vt}^i, \mathbf{M}_m^i) \tag{15}$$

where $\wp_{VMD}(.)$ represents the mapping function of VMD.

Finally, the tracking box with a latency offset is the final predictive tracking result.

$$B_i^p = \varsigma(\mathcal{M}_i, B_i) \tag{16}$$

where $\varsigma(.)$ is the approach to transform the motion and box to the predictive box.

As a result, the predictor can be represented as

$$\begin{aligned} &P((B_i, B_{i-1}, \Delta t, \mathbf{F}_{vt}^i, \mathbf{M}_m^{i-1}) = \\ &\quad \varsigma(\wp_{VMD}(\Delta t, \mathbf{F}_{vt}^i, \mu_{M3}(B_i, B_{i-1}, \mathbf{M}_m^{i-1})), B_i) \end{aligned} \tag{17}$$

4.2. Visual Memory Module

In the Visual Memory Module (VMM), the previous visual memory is input into the cascaded Transformer modules in a hierarchical manner to extract the latest visual memory. It forms visual features with temporal information based on its hierarchical visual features and visual memory.

Specifically, it is assumed that the backbone outputs visual features at three different hierarchical levels, which can be represented as $\mathbf{F}_x^{s1} \in \mathcal{R}^{c1 \times \frac{w_X * h_X}{16}}$, $\mathbf{F}_x^{s2} \in \mathcal{R}^{c2 \times \frac{w_X * h_X}{32}}$ and $\mathbf{F}_x^{s3} \in \mathcal{R}^{c3 \times \frac{w_X * h_X}{64}}$. For each hierarchical feature, there is a corresponding visual memory. For the three different hierarchical levels, the last visual memory $\mathbf{M}_v^{i-1} = [\mathbf{q}_1^{i-1}, \mathbf{q}_2^{i-1}, \mathbf{q}_3^{i-1}]$, where q_n^{i-1} is the visual memory of n-th level in the $i-1$ frame.

Then, three cascaded cross attention are used to update the visual memory from $\mathbf{M}_v^{i-1}$ to $\mathbf{M}_v^i$ as shown in Figure 5. The CA includes a Multi-Head Attention, two Add and Norm layers, and an MLP layer.

The update function can be represented as

$$\mathbf{q}_{12}^i = CA(\mathbf{F}_x^{s3}, \mathbf{q}_1^{i-1}). \tag{18}$$

$$[\mathbf{q}_{123}^i, \mathbf{q}_{23}^i] = CA(\mathbf{F}_x^{s2}, [\mathbf{q}_{12}^i, \mathbf{q}_2^{i-1}]). \tag{19}$$

$$\mathbf{M}_v^i = [\mathbf{q}_1^i, \mathbf{q}_2^i, \mathbf{q}_3^i] = CA(\mathbf{F}_x^{s1}, [\mathbf{q}_{123}^i, \mathbf{q}_{23}^i, \mathbf{q}_3^{i-1}]). \tag{20}$$

where $\mathbf{q}_{12}^i$,$\mathbf{q}_{123}^i$,$\mathbf{q}_{23}^i$ represent the temporary memory for the q_1^{i-1} and q_2^{i-1} after the CA. Finally, the output is the current visual memory $\mathbf{M}_v^i = (\mathbf{q}_1^i, \mathbf{q}_2^i, \mathbf{q}_3^i)$.

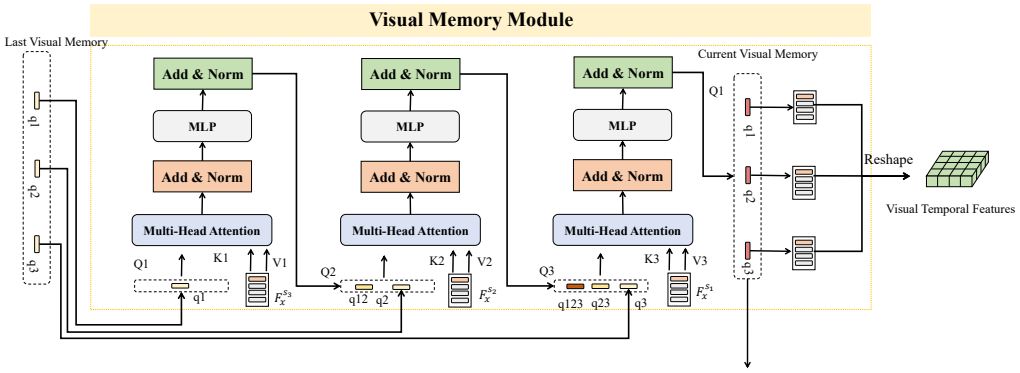


Figure 5. The detailed architecture of the Visual Memory Module.

Then, to obtain the visual temporal features $\mathbf{F}_{vt}$, a simple and effective operation is used to fuse the visual memory and visual features.

$$\mathbf{F}_{vt} = \text{Upsample}(\text{Upsample}(\mathbf{F}_{vt}^{s1}) + (\mathbf{F}_{vt}^{s2}) + (\mathbf{F}_{vt}^{s3})), \tag{21}$$

where $\mathbf{F}_{vt}^{sn}$ means the visual temporal feature in stage $n \in (1, 2, 3)$. For each stage, the features are calculated by the simple Dot product with visual memory.

$$\mathbf{F}_{vt}^{s1} = \mathbf{F}_x^{s3} \odot q1, \tag{22}$$

$$\mathbf{F}_{vt}^{s2} = \mathbf{F}_x^{s2} \odot q2, \tag{23}$$

$$\mathbf{F}_{vt}^{s3} = \mathbf{F}_x^{s1} \odot q3, \tag{24}$$

where $\odot$ means Dot product.

The $\mathbf{F}_{vt}$ will be input into the head and the Predictor.

4.3. Motion Memory Module

The prediction is designed by an offset paradigm as shown in Figure 6. Here, the offset is named as motion, which means the location displacement or shape change between frames. The Motion Memory Module is designed to extract the historical motion memory by fusing the last motion memory M_m^{i-1} and the boxes of the last two frames. However, the simple motion can not describe the offset due to discontinuous frames of latency-aware tracking.

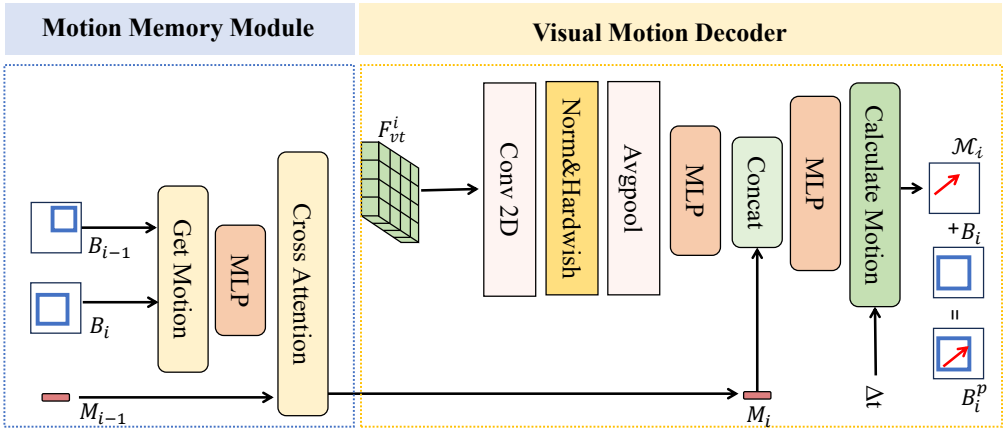


Figure 6. The detailed architecture of the Motion Memory Module and the Visual Motion Decoder.

To represent the motion change rate between continuous frames, the velocity V is proposed. Asuming the tracking boxes of last two frames are $B_i = [x_i, y_i, w_i, h_i]$ and $B_{i-1} = [x_{i-1}, y_{i-1}, w_{i-1}, h_{i-1}]$, the velocity V can be represented as

$$\mathbf{V}_i(B_i, B_{i-1}) = \frac{1}{\triangle f_i}\left[\frac{\triangle_x(f_i)}{w_{f_{i-1}}}, \frac{\triangle_y(f_i)}{h_{f_{i-1}}}, \log\left(\frac{w_f}{w_{f_{i-1}}}\right), \log\left(\frac{h_f}{h_{f_{i-1}}}\right)\right]. \tag{25}$$

where f_i is the latest processed frame, and the f_{i-1} is the last processed frame. $\triangle f_i$ is the frame interval between f_i and f_{i-1}.

$$\triangle f_i = t_g^{f_i} - t_g^{f_{i-1}}. \tag{26}$$

$\triangle_x(f_i)$ and $\triangle_y(f_i)$ are the distance from trackers r_{f_i} and $r_{f_{i-1}}$.

$$\triangle_x(f_i) = x_{f_i} - x_{f_{i-1}}, \tag{27}$$

$$\triangle_y(f_i) = y_{f_i} - y_{f_{i-1}}. \tag{28}$$

The velocity $\mathbf{V}_i$ is the base velocity, and it can only reflect the velocity of the last two frames.

Then the M3 can be represented as

$$\mathbf{M}_m^i = CA(V_i(B_i, B_{i-1}), \mathbf{M}_{i-1}^m). \tag{29}$$

4.4. Visual Motion Decoder

The Visual Motion Decoder (VMD) is designed to fuse visual temporal features $\mathbf{F}_{vt}^i$, the current motion memory $\mathbf{M}_m^i$, and latency Δt to ultimately achieve motion prediction. Introducing the visual temporal features $\mathbf{F}_{vt}^i$ output by the tracker strengthens the coupling relationship between the tracker and the predictor, which is one of the core innovations of this method. The specific structure of the visual visual decoder is shown in Figure 6, and its complete inference process and key details are described step by step as follows.

4.4.1. Explanation of Core Inputs and Memory Mechanism

During the inference process, the core inputs of the decoder and the key information of related memory modules are defined as follows to clarify the function and interaction logic of each component:

Visual temporal features $\mathbf{F}_{vt}^{i}$ is extracted and output by the tracker, it is used to characterize the temporal visual information of the target in the i-th frame, specifically including the trend of the target's appearance changes. This feature is generated by the temporal convolution module of the tracker and updated synchronously with the input of each frame of image, the update frequency is 1 frame/time.

Motion memory $\mathbf{M}_{m}^{i}$ is a query to represent the continuous historical motion trends which is extracted by Motion Memory Module. The update frequency is 1 frame/time.

Latency Δt is the time interval between the "moment when the picture is captured" and the "moment when the predictive tracker outputs the final tracking box B_i^p". Since most of the runtime is spent on input and tracker, we approximately use the moment of entering MVD to replace the moment of outputting the final tracking box. Through the system clock module of the device, record the system timestamp T_1 when the picture is captured and the system timestamp T_2 when the M3 outputs Current Motion Memory respectively, then $\Delta t = |T_2 - T_1|$.

4.4.2. Step-by-Step Inference Process

Firstly, the visual temporal feature $\mathbf{F}_{vt}^{i}$ is input into the conv3d layer, Norm and Hardwish layer, and Avgpool layer. The visual temporal features of the i-th frame $\mathbf{F}_{vt}^{i}$, are sequentially fed into the 3D convolution layer, normalization layer, Hardswish activation layer, and global average pooling layer to complete feature extraction and dimensionality reduction, preparing for fusion with the current motion memory.

$$\mathbf{F} = Avgpool(Norm\&Hardwish(conv3d(\mathbf{F}_{vt}^{i}))) \tag{30}$$

where $\mathbf{F}$ is the temporary feature.

Then, fuse the motion memory feature $\mathbf{M}_{m}^{i}$ with the above temporary visual feature $\mathbf{F}$, and predict the movement speed of the target through a multi-layer perceptron to learn the movement correlation rules of the target.

$$\mathbf{V}_{i}^{p} = MLP(\mathbf{M}_{m}^{i}, \mathbf{F}) \tag{31}$$

where $\mathbf{V}_{i}^{p}$ is the predicted velocity.

By combining the system inference delay Δt and the predicted motion speed, the predicted motion offset of the target in the i-th frame is calculated to compensate for the time delay caused by the inference process.

$$\mathcal{M}_i = \mathbf{V}_{i}^{p} \Delta t, \tag{32}$$

Here, the $\mathcal{M}_i$ is the motion of i-th frame.

Finally, using the obtained predicted motion offset $\mathcal{M}_i$, the position and scale of the initial tracking box B_i of the i-th frame are corrected to obtain the final predicted tracking box B_i^p.

$$B_i^p[0] = B_i[0] + \mathcal{M}_i[0] * B_i[2] \tag{33}$$

$$B_i^p[1] = B_i[1] + \mathcal{M}_i[1] * B_i[3] \tag{34}$$

$$B_i^p[2] = B_i[2] e^{\mathcal{M}_i[2]} \tag{35}$$

$$B_i^p[3] = B_i[3] e^{\mathcal{M}_i[3]} \tag{36}$$

where $X[n]$ represents the n-th element of the X.

During the training process, we hope that the predicted motion bias is as small as possible. Therefore, we calculate the loss between the real bias and the actual bias, and the loss function is:

$$L_M = L_1(\mathcal{M}_i, \widehat{\mathcal{M}_i}), \tag{37}$$

where $\widehat{\mathcal{M}_i}$ is the ground truth of motion.

5. Experiments and Results

In this section, the proposed TPC-Tracker is validated on both simulators and datasets. The visual tracking algorithm is further deployed on a vertical takeoff and landing fixed-wing platform with a flight speed of up to 20 m/s. Experimental results and analyses demonstrate the effectiveness of our method, showcasing its robustness in high-speed motion scenarios and verifying its practical applicability in real-world airborne systems.

5.1. Performance Metrics

5.1.1. Physical Tracking Metrics

To evaluate the TPC-Tracker's latency-aware performance, we adopt the Physical Tracking Trajectory error as the metric in the simulation experiments. Assuming the target trajectory point set in 3-D space is $\{(x_t^{uav}, y_t^{uav}, z_t^{uav})\}$ and the drone trajectory point set is $\{(x_t^{uav}, y_t^{uav}, z_t^{uav})\}$, the two are strictly aligned in time (N time points in total). The following are the calculation formulas for mean square error MSE:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}\left[(x_t^{uav} - x_t^{target})^2 + (y_t^{uav} - y_t^{target})^2\right] \tag{38}$$

5.1.2. Offline Metrics

To evaluate the TPC-Tracker accuracy performance under the offline evaluation, we adopt normalized distance precision [36] ($Norm.P$), which is founded on the center location error (CLE), and the area under the curve (AUC), which is based on the intersection over the union. The specific calculation formula is as follows:

$$Norm.CLE = \frac{\sqrt{(x^{pred} - x^{gt})^2 + (y^{pred} - y^{gt})^2}}{\sqrt{w^{gt}h^{gt}}} \tag{39}$$

where x, y, w, h is the tracking box.

For a given threshold τ, if the $Norm.CLE$ of a certain frame is less than or equal to τ, the tracking of that frame is considered "successful".

$$F(x,\tau)\begin{cases}1 & \text{if x}\leq \tau\\ 0 & \text{if x}> \tau\end{cases} \tag{40}$$

$$Norm.P = \frac{\sum_{i=1}^{N} F(Norm.CLE_i, \tau)}{N}, \tag{41}$$

where N is the total number of frames.

For each frame tracking result, the overlap rate OR is defined as:

$$OR = \left|\frac{B_{track} \cap B_{gt}}{B_{track} \cup B_{gt}}\right| \tag{42}$$

where the B is the tracking box.

For a given threshold δ, if the OR of a certain frame is more than or equal to δ, the tracking of that frame is considered "successful".

$$G(x,\delta)\begin{cases}1 & \text{if } x > \delta \\ 0 & \text{if } x \leq \delta\end{cases} \tag{43}$$

$$SR(\delta) = \frac{\sum_1^N G(OR)}{N}, \tag{44}$$

Calculate the success rates of all thresholds within the interval $delta \in [0,1]$ at a certain step size, such as $delta$ = 0, 0.05, 0.1, 1, obtain discrete points, and form a continuous curve through interpolation or direct connection.

AUC is the area of the curve in the interval of $delta \in [0,1]$, which is usually approximated using the trapezoidal method:

$$AUC = \sum_{i=1}^{n} \frac{(\delta_i - \delta_{i-1})(SR(\delta_i) - SR(\delta_{i-1}))}{2}, \tag{45}$$

5.1.3. Latency-Aware Metrics

For LAE [11], the principle has already been described in the Preliminary. We define the AUC and $Norm.Prec$ under LAE as the $AUC@La$ and $Norm.Prec$.

$$AUC@La\Delta = LAE(AUC), \tag{46}$$

$$Norm.Prec@La\Delta = LAE(Norm.Prec), \tag{47}$$

Note that the latency-aware metric is a mixed metric, which not only reflects the accuracy but also reflects the inference speed, because it has already punished the slow inference speed model.

5.2. Implementation Details

5.2.1. Dataset

TPC-Tracker is trained on TrackingNet [37], COCO [38], LaSOT [39] and Got10k [40]. The evaluation takes authoritative UAV tracking datasets, UAV123 [16], visdrone [17], and Guidance-UAV [8]. The Guidance-UAV is a dataset that focuses on visual guidance, which is extremely sensitive to time delay.

5.2.2. Model Variants

We develop three variants of TPC-Tracker models with different lightweight transformers. We adopt LeViT-256, LeViT-128, and LeViT-128S for TPC-Tracker-Large, TPC-Tracker-base, and TPC-Tracker-small, respectively. In addition, Table 1 reports model parameters, FLOPs, and inference speed on multiple devices. The platforms used for inference are GPU Nvidia RTX 4090, NVIDIA ORIN NX and RK3588S.

Table 1. Details of our TPC-Tracker model variants.

Model		TPC-Base	TPC-Large	TPC-Small
PyTorch Speed (fps)	GPU	97	81	105
	NX	24	21	26
ONNX Speed (fps)	GPU	102	77	105
	NX	25	21	27
RKNN Speed (fps)	RK3588S	12	8	15
Flops (G)		4.98	8.76	2.35
Params (M)		51.33	92.71	22.84

5.2.3. Training Strategy

TPC-Tracker is trained using the datasets TrackingNet, COCO, LaSOT, and Got10k. The network takes an image pair as input, which consists of a template image and a search image. In the case of video datasets, the image pair is sampled from a randomly chosen video sequence. For the image dataset COCO, an image is randomly selected, and then data augmentations are applied to generate an image pair. Common data augmentation techniques like scaling, translation, and jittering are employed on the image pair. The search region is obtained by expanding the target box by a factor of 4, while the template is obtained by expanding it by a factor of 2. Subsequently, the search image is resized to 256 × 256, and the template image is resized to 128 × 128. The transformer in TPC-Tracker is initialized with ImageNet pre-trained LeViT, and the remaining parameters of TPC-Tracker are initialized randomly. The AdamW optimizer is used, with a weight decay of 1×10^{-4}. The initial learning rate of TPC-Tracker is set to 3×10^{-4}. To train the model, 4 Nvidia RTX 4090 GPUs are utilized. The training process runs for 1500 epochs with a batch size of 64. Each epoch includes 60,000 sampling pairs.

5.3. Rflysim Simulator Remote Sensing Experiments

5.3.1. Experiments Setup

To verify the effectiveness of our algorithm, we conducted a remote sensing tracking task in the Rflysim simulator [41]. This is a hardware-in-loop experiment. The video stream is captured from the simulator and input into the hardware Jetson Orin NX, where TPC-Tracker performs inference on the video stream. The hardware inference results and tracking control commands are then input back into the flight control system Coptersim. Under the same control strategy, the trajectory error between the moving target and the UAV varies with different trackers. The experimental method was as Figure 7 shows: The UAV physically tracked the target person at a height of 20 m. The control algorithm was designed to keep the height unchanged and always keep the person in the center of the field of view.



(a) (b)

Figure 7. Schematic diagrams of the Rflysim experimental scene. (**a**) shows the third-person view, where the UAV and the target person are framed, and the target person is moving in a curved path. (**b**) shows the first-person view of the UAV.

Four moving target trajectories are designed: linear motion, curvilinear motion to test the capabilities of trackers. We will compare the high-speed algorithm SiamRPN [21], the high-precision algorithm TATrack [1], and two predictive trackers (PVT [11] and PVT++ [12]) as benchmarks.

5.3.2. Result and Analysis

In Table 2, the Mean Squared Error (MSE) results and FPS of different trackers tracking targets under different devices are presented. The following conclusions can be drawn from the results:

(1) Trackers with slow inference speed and no prediction capability exhibit the worst physical tracking performance. TCTracker, for example, achieves an FPS as low as 8 on Orin NX 8G and shows severe MSE values: 8.91 for Linear motion and 92.78 for Curvilinear motion, indicating its inability to maintain effective tracking due to sluggish response.

(2) When computing power is relatively sufficient, predictive trackers demonstrate stronger tracking performance. PVT and PVT++, which incorporate prediction mechanisms, achieve lower MSE values. On Orin NX 8G, PVT++ records 1.76 for Linear motion and 3.80 for Curvilinear motion, outperforming non-predictive counterparts like SiamRPN, particularly in dynamic scenarios.

(3) Algorithms with similar accuracy can produce entirely different tracking results on different devices due to FPS variations, highlighting the need to introduce delay-aware evaluation. SiamRPN achieves an FPS of 73 on Orin NX 8G but only 24 on RK3588S, leading to significantly higher MSE 6.06 for Curvilinear motion, on the latter device due to increased latency.

(4) The proposed method (TPC-Tracker) achieves the best physical tracking performance, with the lowest MSE values across devices. On Orin NX 8G, it records 1.12 for Linear motion and 2.70 for Curvilinear motion, combined with balanced FPS, demonstrating its superiority in real-world tracking tasks.

Table 2. The Mean Squared Error (MSE) results of different trackers tracking targets under different devices.

Method	Device AUC	Orin NX 8G			Xavier NX 16G			RK3588S		
		Linear	Curvilinear	FPS	Linear	Curvilinear	FPS	Linear	Curvilinear	FPS
SiamRPN	58.40	2.88	4.51	73	2.88	4.51	53	4.01	6.06	24
TATracker	64.12	2.67	4.43	38	2.60	4.43	31	4.17	6.30	19
TCTracker	64.34	8.91	92.78	8	9.11	114.37	5	–	–	–
PVT	60.34	1.95	4.01	15	2.59	4.44	12	3.97	6.14	10
PVT++	60.34	1.76	3.80	14	2.53	4.38	12	3.62	5.99	10
TPC-Tracker	63.52	1.12	2.70	25	2.18	3.78	22	3.04	5.47	12

The top result is shown in red color.

5.4. Complexity Analysis

5.5. Comparisons Under Latency-Aware Evaluation

To demonstrate the necessity of latency awareness, we evaluated the performance of 14 common trackers under both normal evaluation and latency-aware evaluation. The baselines are divided into two groups: Normal Tracker and Latency-aware Tracker. For the former, there are SiamFC [20], SiamRPN [21], Siammask [23], MixformerV2 [42], Hit [9], TCTracker [35], ARTracker [43], DeconNet [44], SeqTrack [45], Mobilesiam-ST [46], and TATracker-Deit [1]. For the latter, there are PVT [11] and PVT++ [12].

We add a symbol *@La* after the metric to indicate that the metric is under latency-aware evaluation. For fairness, for all methods, the conventional metrics are tested on the NVIDIA RTX 4090, while the latency-aware metrics are tested on the NVIDIA ORIN NX. The top result is shown in red text. The second is shown in blue text.

UAV123. UAV123 [16] is constructed for low-altitude UAVs remote sensing, which enables it to capture a wide variety of real-world scenarios. This dataset contains 123 video clips, covering diverse environments such as urban areas, rural landscapes, and natural settings.

As shown in Table 3, it is evident that TPC-Tracker-Base stands out among all the trackers in terms of achieving the best *AUC@La* performance at 61.42 on the UAV123 dataset. While the Mixformer attains a higher *AUC* value at 70.41, which reflects its overall tracking accuracy in offline tracking, the *AUC@La* is only 52.1. Comparing the *AUC@La* and *AUC*, the decrease rate reflects the gap of inference latency. The TPC-Tracker-Tiny has

the smallest decline range under Latency-Aware Evaluation compared with the normal *AUC*, which only decreases by 0.4%.

Table 3. Performance comparisons with state-of-the-art trackers on the test set of UAV123, Visdrone and Guidance-UAV. The subscripts represent the decline ratio of LAE compared to the normal evaluation. The metrics with La are the Latency-aware metrics, which can really reflect the ability to solve the latency.

Type	Method	Source	UAV123		Visdrone				Guidance-UAV		Speed (fps)
			AUC	AUC@La$_\Delta$	AUC	AUC@La$_\Delta$	Norm.P	Norm.P@La$_\Delta$	AUC	AUC@La$_\Delta$	Orin NX 8G
Normal Trackers	SiamFC [20]	ECCV16	46.80	43.99$_{-6\%}$	48.54	45.52$_{-6\%}$	54.52	51.04$_{-6\%}$	43.79	40.73$_{-7\%}$	39
	SiamRPN [21]	CVPR18	58.40	55.48$_{-5\%}$	64.19	60.66$_{-5\%}$	67.23	63.06$_{-6\%}$	53.01	51.42$_{-3\%}$	73
	Siammask [23]	CVPR19	60.34	52.50$_{-13\%}$	65.61	56.34$_{-14\%}$	72.05	60.88$_{-15\%}$	54.64	47.53$_{-13\%}$	16
	OSTrack [10]	ECCV22	68.30	58.06$_{-15\%}$	73.76	61.81$_{-16\%}$	81.04	64.47$_{-18\%}$	63.94	52.44$_{-18\%}$	13
	MixformerV2 [42]	NEURIPS23	70.41	52.10$_{-26\%}$	74.07	53.81$_{-27\%}$	80.48	56.89$_{-29\%}$	63.89	48.56$_{-24\%}$	7
	Hit [9]	ICCV23	58.34	55.42$_{-5\%}$	58.53	55.59$_{-5\%}$	61.54	58.10$_{-6\%}$	54.97	51.90$_{-6\%}$	39
	TCTrack [35]	CVPR22	64.34	52.76$_{-18\%}$	68.61	55.44$_{-19\%}$	73.57	57.93$_{-21\%}$	61.07	47.02$_{-23\%}$	8
	ARTrack [43]	CVPR23	66.34	53.07$_{-20\%}$	69.35	54.85$_{-21\%}$	74.63	58.25$_{-22\%}$	63.00	49.17$_{-22\%}$	6
	SeqTrack [45]	CVPR23	68.60	56.25$_{-18\%}$	70.61	57.53$_{-19\%}$	75.11	59.31$_{-21\%}$	62.14	47.85$_{-23\%}$	9
	DeconNet [44]	TGRS22	60.16	57.34$_{-6\%}$	62.86	58.77$_{-7\%}$	59.37	56.29$_{-5\%}$	69.49	64.47$_{-7\%}$	21
	Mobilesiam-ST [46]	TGRS24	61.20	-	61.10	-	-	-	-	-	19
	TATracker-Deit [1]	TGRS24	64.12	60.27$_{-6\%}$	72.31	68.67$_{-5\%}$	70.79	67.96$_{-4\%}$	59.22	55.67$_{-6\%}$	38
Trackers & Predictor	PVT(SiamMask) [11]	arXiv21	60.34	57.93$_{-4\%}$	61.34	58.84$_{-4\%}$	67.00	63.90$_{-5\%}$	56.99	54.36$_{-5\%}$	15
	PVT++(SiamMask) [12]	ICCV23	60.34	58.53$_{-3\%}$	65.45	63.30$_{-3\%}$	67.58	65.23$_{-3\%}$	57.00	56.02$_{-3\%}$	14
	TPC-Track-Base	Ours	63.32	61.42$_{-2\%}$	72.34	70.06$_{-3\%}$	72.41	70.96$_{-2\%}$	58.08	56.92$_{-2\%}$	25
	TPC-Track-Large	Ours	63.73	60.40$_{-5\%}$	73.02	69.53$_{-5\%}$	73.23	70.22$_{-4\%}$	58.82	56.92$_{-3\%}$	21
	TPC-Track-Tiny	Ours	60.59	60.34$_{-0.4\%}$	72.34	69.14$_{-4\%}$	69.92	67.08$_{-4\%}$	55.08	54.04$_{-1\%}$	27

Visdrone-SOT. The VisDrone-SOT [17] dataset consists of 288 video clips formed by 261,908 frames and 10,209 static images. These data are captured by various drone-mounted cameras, covering a wide range of aspects and different conditions. It is also suitable for low-altitude remote sensing.

As indicated in Table 3, the results are nearly the same as UAV123. Mixformer achieves the best *AUC* and *Norm.P* score under the normal evaluation but the *AUC@La* and *Norm.P@La* decline 27% and 29%, respectively. The reason is that Mixformer only infers images at 7 FPS on Jetson Orin NX. The TPC-Tracker-Base achieves the *AUC@La* and *Norm.P* respectively at 70.06% and 70.96%. The decrease rate is lower than the TPC-Tracker-Tiny's though the fps is lower.

Guidance-UAV. The Guidance-UAV [8] dataset which focuses on the visual guidance for UAVs, consists of 16 video clips formed by 3209 frames.

As shown in Table 3, different trackers exhibit varying performance on this dataset. For instance, OsTrack achieves an *AUC* of 63.94% and an *AUC@La* of 52.44% with a decline rate of 0.18. This indicates its robustness in handling the complex visual cues and time-sensitive requirements of the Guidance-UAV. The TPC-Tracker-Base and TPC-Tracker-Large are both the best trackers on *AUC@La*. Meanwhile, the lowest decline rate is TPC-Tracker-Tiny.

Speed According to the result in Table 3. The speed of trackers without predictors determined the decline rate for latency-aware evaluation. When the FPS is larger than $\kappa = 30$, the decline rate is about 5%, cause the results have only one frame latency. When the FPS is smaller than $\kappa = 30$, the smaller the FPS the larger the decline rate. It causes some trackers with high accuracy can not be used in real-time tracking like Mixformer.

Note. The Latency-aware metric and FPS only represent the tracker tested on the Nvidia Jetson Orin NX 8G. When the device is changed, the FPS and Latency-aware metrics are changed too.

5.6. Visualization on Predictive Tracking

To further understand the TPC-Tracker, as Figure 8 shows, we visualize the tracking results of Latency trackers including TPC-Tracker, PVT, and PVT++ in Guidance-UAV. Our tracker is designed to capture not only visual temporal features but also global target

motion information, going beyond just short-term motion. This unique capability allows for notably more accurate tracking, particularly when dealing with large and abrupt motion sequences.

As shown in Figure 8, in the early stage of the sequences, the performance of the three predictive trackers is similar to that of the ground truth. This is because the object is far away from the UAV, and the relative motion within the frame is slow. However, in the later stage, it is different. The PVT is the worst method because it simply uses the Kalman Filter without taking the temporal features into account. Apparently, its tracking result is just similar to that of the last frame which is small and offset. Our TPC-Tracker stands out among all the methods as it can model the entire motion process.

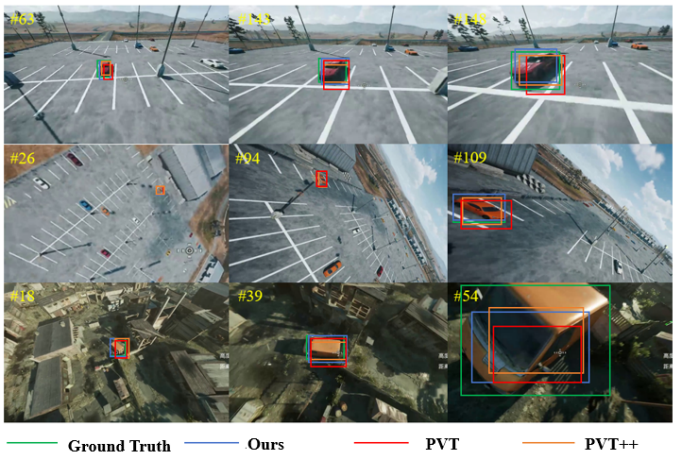


Figure 8. The Visualization of Ground Truth, TPC-Tracker, PVT and PVT++ in Guidance-UAV.

5.7. Runtime Breakdown

As shown in Table 4, the Tracker accounts for the largest proportion (63.8%, 25.67 ms/frame) due to global temporal feature encoding and fusion. The Predictor takes 7.12 ms/frame (17.7%), while Input Preprocessing (4.63 ms/frame, 11.5%) and Post-processing (2.82 ms/frame, 7.0%) are lightweight. The total runtime is 40.23 ms/frame (25 FPS), meeting UAV real-time tracking requirements. The Tracker is the key for future optimization.

Table 4. Runtime breakdown of the TPC-tracker-base on NVIDIA ORIN NX.

Module	Average Time per Frame (ms)	Proportion of Total Runtime (%)
Input Preprocessing	4.63	11.5
Tracker	25.67	63.8
Predictor	7.12	17.7
Post-processing	2.82	7.0
Total Runtime	**40.23**	**100.0**

5.8. Ablation Study and Analysis

5.8.1. Components Analysis

To verify the effectiveness of each module in our tracker, we removed the Visual Memory Module (VMM) and the M3 respectively on the basis of TPC-Tracker-Base. The result is shown in Table 5. The VMM can significantly improve the *AUC* of target tracking, yet it cannot reduce the decline ratio. The Predictor cannot improve the *AUC*, but it can enhance the *AUC@La* by reducing the decline ratio.

In the case of the UAV123 dataset, when only the Predictor is added, while the AUC remains the same as the baseline without any module addition, the AUC@La shows a

notable improvement, with the decline ratio decreasing from −0.05 to −0.02. This clearly demonstrates the Predictor's capacity to better handle the target's long-term associations and mitigate the performance degradation over time, even if it does not directly boost the initial AUC value.

Table 5. The impact of VMM and M3 on UAV123 and Guidance-UAV.

	Components		UAV123		Guidance-UAV	
	VMM	**M3**	**AUC**	**AUC@La$_\Delta$**	**AUC**	**AUC@La$_\Delta$**
1			58.34	$55.42_{-5\%}$	54.97	$51.90_{-6\%}$
2		True	58.34	$56.97_{-2\%}$	54.97	$52.60_{-4\%}$
3	True		63.32	$58.66_{-6\%}$	55.00	$50.05_{-3\%}$
4	True	True	63.32	$61.42_{-2\%}$	55.08	$54.04_{-1\%}$

On the other hand, when only the VMM is incorporated, we observe a significant jump in the *AUC* from 58.34 to 63.32, validating its crucial role in capturing essential visual temporal cues that enhance the tracker's accuracy in object identification and localization. However, as expected, the decline ratio remains relatively stable at −0.06, indicating that while it excels in improving the initial accuracy, it does not inherently possess the ability to counteract the subsequent performance drop as effectively as the M3.

When both the VMM and the M3 are combined, we achieve the best of both worlds. The *AUC@La* reaches an impressive 61.42 with a decline ratio of just −0.02 and a solid *AUC* of 63.32. This synergy showcases how the two modules complement each other, with the VMM laying the foundation for accurate initial tracking and the Predictor ensuring stability and enhanced performance as the sequence progresses.

A similar trend can be observed in the Guidance-UAV dataset. The addition of the Predictor leads to an improvement in *AUC@La* and a reduction in the decline ratio, while the VMM boosts the overall *AUC*. Once again, the combination of both modules yields the optimal performance metrics, underlining the importance of carefully designed and integrated components in developing a highly effective target tracker for diverse UAV applications. Future research could focus on further optimizing these modules or exploring additional features that could potentially enhance the tracker's adaptability and precision in even more challenging scenarios during remote sensing tasks.

5.8.2. Experiments on Different Latency

Different cameras on the UAVs have different fps speeds when performing different remote sensing tasks. As a result, this paper conducted experiments to explore the results of predictive tracking at different speeds κ, which is the fps captured by the camera. The tracker is tested when the speed κ is 10, 20, 30, 40, 50, 60.

The specific approach is as follows: one thread plays at a speed of κ, and the other thread runs TPC-Tracker. The input of TPC-Tracker is the picture being played at the current moment, and the result of the current picture is the previous result output by the tracker at the moment when the picture started playing. There is no need for deliberate alignment.

As shown in Table 6, this paper systematically conducted a series of experiments with the aim of exploring the results of predictive tracking under different speeds, denoted as, which specifically refers to the frames per second (fps) captured by the camera. The tracker's performance was meticulously tested when the speed was set to 10, 20, 30, 40, 50, and 60 fps, respectively.

Table 6. The influence of κ to predictor.

	UAV123		Guidance-UAV	
	AUC	**AUC@La$_\Delta$**	**Norm.P**	**Norm.P@La$_\Delta$**
10	63.32	$62.68_{-1\%}$	58.08	$56.95_{-2\%}$
20	63.32	$61.43_{-2\%}$	58.08	$56.93_{-2\%}$
30	63.32	$61.42_{-2\%}$	58.08	$56.92_{-2\%}$
40	63.32	$60.79_{-4\%}$	58.08	$56.34_{-3\%}$
50	63.32	$60.75_{-4\%}$	58.08	$55.18_{-5\%}$
60	63.32	$60.73_{-4\%}$	58.08	$55.17_{-5\%}$

As clearly shown in Table 6, which presents the detailed findings of this investigation, we can observe several notable trends. In the case of the UAV123 dataset, when the speed was at the relatively lower value of 10 fps, the tracker achieved an AUC of 63.32 and an AUC@La of 62.68 with a decline rate of only -0.01. This indicates a relatively stable performance in the initial stages of tracking, suggesting that the tracker was able to effectively capture and follow the target with minimal deviation. As the speed κ increased to 20 fps, the *AUC* remained consistent at 63.32, while the *AUC@La* slightly decreased to 61.43 with a of -0.02.

When the κ is larger than the inference speed, the LAE still only decreases by 0.05. This indicates that the prediction module can effectively alleviate the problem of latency perception. It implies that the prediction module has a significant role in compensating for the potential negative impacts caused by the delay between the actual speed and the inference speed. Making reasonable predictions and adjustments helps to maintain a relatively stable performance level, reducing the performance degradation to a minimal extent even when facing such challenging situations where the speed exceeds the inference speed. This showcases the effectiveness and importance of the prediction module in enhancing the overall adaptability and reliability of the tracking system.

5.9. Real-World Remote Sensing Experiments

We also carried out real-world tests to further validate the practicality and robustness of our algorithm. The test site was a civilian airport, which provided a real-world and complex environment with various potential interferences such as wind, other flying objects, and ground structures. The experimental platform was a vertical takeoff and landing (VTOL) fixed-wing aircraft. This type of aircraft combines the advantages of vertical takeoff and landing like a helicopter and the high-efficiency long-distance flight performance of a fixed-wing aircraft. During the tests, the aircraft flew at an altitude of 70 m with an airspeed of 20 m per second, simulating high-speed and long-range flight scenarios that are common in practical applications. The network camera equipped on the aircraft used the H264 encoding format. To save transmission bandwidth, we set the bit rate to the lowest level. However, this decision led to some loss during the transmission and decoding process, resulting in blurred images. This real-world challenge of image quality degradation is typical in many practical scenarios, making our tests more representative. The results of the target tracking are presented in the following Figure 9.

The proposed TPC-Tracker-Base demonstrates superior tracking performance across all metrics in real-world scenarios. As shown in Table 7, it achieves state-of-the-art results with 62.11% AUC and 58.41% Norm.P, outperforming PVT++ by 1.12% and 1.93%, respectively. Notably, PVT++ exhibits a 0.91% AUC and 2.24% Norm.P improvement over its baseline PVT, indicating the effectiveness of algorithmic refinements in intermediate versions. Despite the blurred images caused by the low-bit-rate encoding and the high-speed movement of the aircraft, TPC-Tracker can still accurately predict and track the target in the

aerial images captured by the high-speed moving UAV in the real world. Compared with the previous simulator tests, the real-world tests introduced more uncertainties and challenges. For example, the real-world wind conditions affected the stability of the aircraft's flight, and the complex ground environment increased the difficulty of target recognition. However, TPC-Tracker showed excellent adaptability and robustness. It maintained a high-level tracking accuracy, which is crucial for practical applications such as search and rescue, surveillance, and inspection. In conclusion, through both the simulator tests and the real-world tests, we have fully demonstrated that TPC-Tracker is a highly effective and practical target-tracking algorithm. It can not only perform well in a controlled simulation environment but also show outstanding performance in complex and harsh real-world scenarios, providing a reliable solution for UAV-based target tracking applications.

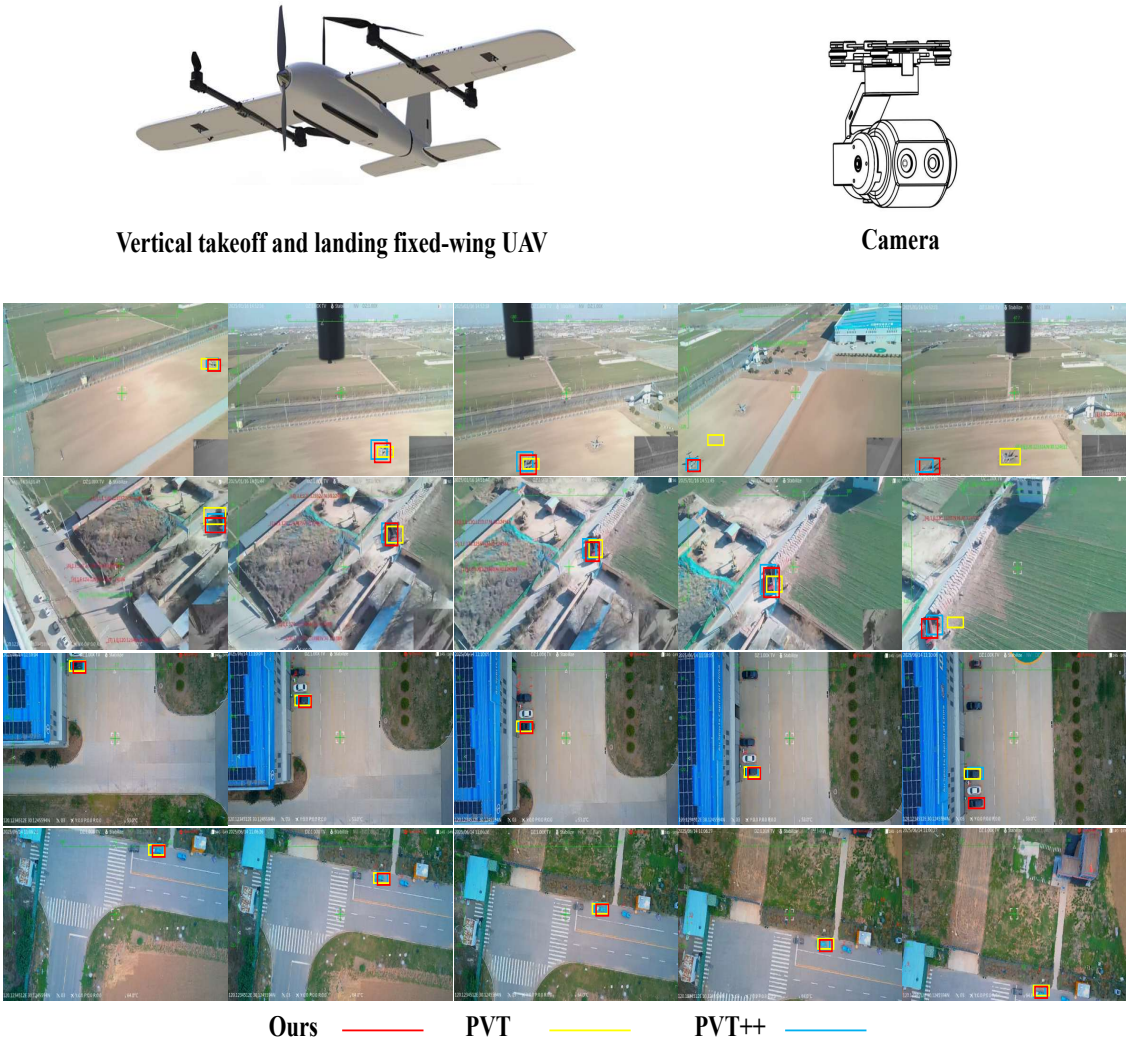


Figure 9. The experiment of our method in the real-world test.

Table 7. The experiment result in realworld.

Method	AUC	AUC@La$_{\Delta}$	Norm.P	Norm.P@La$_{\Delta}$
PVT	60.08	59.48$_{-1\%}$	54.24	53.16$_{-2\%}$
PVT++	60.99	59.77$_{-2\%}$	56.48	55.35$_{-2\%}$
TPC-Tracker-Base	62.11	60.86$_{-2\%}$	58.41	57.83$_{-1\%}$

6. Discussion

6.1. Limitations

In this work, both predictor and the tracker contribute to object state prediction for remote sensing. Their parallel design causes redundant computation during independent feature extraction of remote sensing data, conflicting with UAVs' limited computing power. Direct prediction via images and motion instead of "tracking-then-prediction" can significantly reduce computation. Current algorithms handle extreme scenarios like high-speed targets and large UAV payload attitude changes, yet scalability in complex environments and robustness in harsher motion situations, such as abrupt target acceleration and sharp turns, still need improvement. The proposed direct prediction scheme using images and motion cues requires verification regarding the trade-off between computational efficiency and tracking precision. Future research should establish diverse test systems, develop adaptive end-to-end architectures, and integrate model compression to promote lightweight and robust solutions for UAV remote sensing.

6.2. Prospects

Latency-aware perception is a core component of remote sensing embodied intelligence. Real-time environmental perception capabilities not only ensure efficient interaction between UAVs and the environment but also serve as a common requirement in fields such as robotics and autonomous vehicles. Currently, the evaluation systems and framework methods for latency-aware perception still have significant room for improvement. Addressing this issue will promote the deep integration of perception and control—a conclusion applicable to various intelligent systems including UAVs and robots. Current systems often treat these two modules in isolation, resulting in suboptimal performance in dynamic scenarios. Future research should explore joint optimization frameworks, incorporate control constraints into perception models, and develop real-time collaborative design methods for sensor deployment, perception algorithms, and motion planning to enhance the cross-scenario adaptability of the technology.

7. Conclusions

This paper addresses the latency issue in UAV-based remote sensing aerial tracking—a core challenge for real-time remote sensing tasks like dynamic target monitoring and emergency response—and proposes the TPC-Tracker framework. This framework resolves the core shortcomings of existing predictive tracking methods in remote sensing scenarios, namely the weak correlation between trackers and predictors and the discontinuous modeling of historical information.

Theoretically, its innovations for remote sensing applications lie in two aspects: first, constructing a cross-module feature fusion paradigm through the Visual Motion Decoder (VMD), breaking the independent module mode. This paradigm effectively fuses high-resolution spatial features and dynamic motion features, providing a universal interaction approach for multi-module visual perception in remote sensing tracking. Second, realizing the continuous memory modeling of discrete historical information with the help of the Visual Memory Module (VMM) and Motion Memory Module (M3). This enables efficient retention of long-term spatial–temporal correlations in remote sensing data, offering a new path for time-series dependent remote sensing tasks such as dynamic target trajectory prediction.

Experimental results show that the TPC-Tracker exhibits outstanding performance in remote sensing-oriented UAV tracking: in physical tracking simulations for remote sensing scenarios, the Mean Squared Error (MSE) is reduced by up to 38.95%; in the latency-aware evaluation on three major datasets, including UAV123, the AUC@La outperforms that

of 14 mainstream trackers. In real-world remote sensing tasks, the lowest performance degradation rate is only 0.4%, and Vertical Take-Off and Landing (VTOL) UAVs equipped with this framework can achieve stable tracking of ground dynamic targets at an altitude of 80 m and a speed of 20 m/s.

Author Contributions: Conceptualization, X.Y. and N.Z.; methodology, X.Y.; software, X.Y.; validation, X.Y., Y.X., R.S. and T.W.; formal analysis, X.Y. and T.W.; investigation, X.Y.; resources, X.Y.; data curation, X.Y.; writing—original draft preparation, X.Y.; writing—review and editing, X.Y.; visualization, X.Y.; supervision, N.Z.; project administration, N.Z.; funding acquisition, N.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China, Grant 2022YFB3902304.

Data Availability Statement: Data derived from public domain resources.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, S.; Yang, X.; Wang, X.; Zeng, D.; Ye, H.; Zhao, Q. Learning target-aware vision transformers for real-time uav tracking. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 4705718. [CrossRef]
2. He, B.; Wang, F.; Wang, X.; Li, H.; Sun, F.; Zhou, H. Temporal context and environment-aware correlation filter for uav object tracking. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5630915. [CrossRef]
3. Zhang, N.; Ni, S.; Chen, L.; Wang, T.; Chen, H. High-throughput and energy-efficient fpga-based accelerator for all adder neural networks. *IEEE Internet Things J.* **2025**, *12*, 20357–20376. [CrossRef]
4. Xue, Y.; Jin, G.; Shen, T.; Tan, L.; Wang, N.; Gao, J.; Wang, L. Smalltrack: Wavelet pooling and graph enhanced classification for UAV small object tracking. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5618815. [CrossRef]
5. Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; Yan, J. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 4277–4286.
6. Zhang, Z.; Peng, H. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 4586–4595.
7. Chen, X.; Yan, B.; Zhu, J.; Wang, D.; Yang, X.; Lu, H. Transformer tracking. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 8122–8131.
8. Yang, X.; Tan, Q.; Su, H.; Tan, H. Guidance-tracker: An adaptive drone siamese tracker for visual guidance. *Acta Armamentarii* **2025**, *46*, 240284.
9. Kang, B.; Chen, X.; Wang, D.; Peng, H.; Lu, H. Exploring lightweight hierarchical vision transformers for efficient visual tracking. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 9578–9587.
10. Ye, B.; Chang, H.; Ma, B.; Shan, S.; Chen, X. Joint feature learning and relation modeling for tracking: A one-stream framework. In *Computer Vision-ECCV 2022-17th European Conference, Tel Aviv, Israel, 23–27 October 2022*; Springer: Cham, Switzerland, 2022; Volume 13682, pp. 341–357.
11. Li, B.; Li, Y.; Ye, J.; Fu, C.; Zhao, H. Predictive visual tracking: A new benchmark and baseline approach. *arXiv* **2021**, arXiv:2103.04508.
12. Li, B.; Huang, Z.; Ye, J.; Li, Y.; Scherer, S.; Zhao, H.; Fu, C. Pvt++: A simple end-to-end latency-aware visual tracking framework. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 9972–9982.
13. Yang, K.; Quan, Q. An autonomous intercept drone with image-based visual servo. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 2230–2236.
14. Liu, Y.; Li, R.; Cheng, Y.; Tan, R.T.; Sui, X. Object tracking using spatio-temporal networks for future prediction location. In *Computer Vision-ECCV 2020-16th European Conference, Glasgow, UK, 23–28 August 2020*; Springer: Cham, Switzerland, 2020; Volume 12367, pp. 1–17.
15. Liang, N.; Wu, G.; Kang, W.; Wang, Z.; Feng, D.D. Real-time long-term tracking with prediction-detection-correction. *IEEE Trans. Multimedia* **2018**, *20*, 2289–2302. [CrossRef]

16. Mueller, M.; Smith, N.; Ghanem, B. A benchmark and simulator for uav tracking. In *Computer Vision–ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 445–461.
17. Du, D.; Zhu, P.; Wen, L. Visdrone-det2019: The vision meets drone object detection in image challenge results. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 213–226.
18. Bolme, D.S.; Beveridge, J.R.; Draper, B.A.; Lui, Y.M. Visual object tracking using adaptive correlation filters. In *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010*; IEEE: Piscataway, NJ, USA, 2010; pp. 2544–2550.
19. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 583–596. [CrossRef] [PubMed]
20. Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H.S. Fully-convolutional siamese networks for object tracking. In *Computer Vision–ECCV 2016 Workshops*; Hua, G., Jégou, H., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 850–865.
21. Li, B.; Yan, J.; Wu, W.; Zhu, Z.; Hu, X. High performance visual tracking with siamese region proposal network. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018*; Computer Vision Foundation/IEEE Computer Society: Piscataway, NJ, USA, 2018; pp. 8971–8980.
22. Guo, D.; Wang, J.; Cui, Y.; Wang, Z.; Chen, S. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 6268–6276.
23. Hu, W.; Wang, Q.; Zhang, L.; Bertinetto, L.; Torr, P.H. Siammask: A framework for fast online object tracking and segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 3072–3089. [PubMed]
24. Cui, Y.; Jiang, C.; Wang, L.; Wu, G. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 13598–13608.
25. Li, J.; Xu, M.; Chen, H.; Liu, W.; Chen, L.; Xie, Y. Spatio-temporal pruning for training ultra-low-latency spiking neural networks in remote sensing scene classification. *Remote Sens.* **2024**, *16*, 3200. [CrossRef]
26. Duan, D.; Liu, P.; Hui, B.; Wen, F. Brain-inspired online adaptation for remote sensing with spiking neural network. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 5605818. [CrossRef]
27. Pang, Y.; Yao, L.; Luo, Y.; Dong, C.; Kong, Q.; Chen, B. Repsvit: An efficient vision transformer based on spiking neural networks for object recognition in satellite on-orbit remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5610916. [CrossRef]
28. Yang, S.; Linares-Barranco, B.; Wu, Y.; Chen, B. Self-supervised high-order information bottleneck learning of spiking neural network for robust event-based optical flow estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 2280–2297. [CrossRef] [PubMed]
29. Chen, Y.; Wang, L. eMoE-tracker: Environmental moe-based transformer for robust event-guided object tracking. *IEEE Robot. Autom. Lett.* **2025**, *10*, 1393–1400. [CrossRef]
30. Dionigi, A.; Felicioni, S.; Leomanni, M.; Costante, G. D-VAT: End-to-end visual active tracking for micro aerial vehicles. *IEEE Robot. Autom. Lett.* **2024**, *9*, 5046–5053. [CrossRef]
31. Liang, M.; Yang, B.; Zeng, W.; Chen, Y.; Hu, R.; Casas, S.; Urtasun, R. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 11550–11559.
32. Zhang, Y.; Sun, P.; Jiang, Y.; Yu, D.; Weng, F.; Yuan, Z.; Luo, P.; Liu, W.; Wang, X. Bytetrack: Multi-object tracking by associating every detection box. In *Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2022*; Springer: Cham, Switzerland, 2022; Volume 13669, pp. 1–19.
33. Rudenko, A.; Palmieri, L.; Herman, M.; Kitani, K.M.; Gavrila, D.M.; Arras, K.O. Human motion trajectory prediction: A survey. *Int. J. Robot. Res.* **2019**, *39*, 895–935. [CrossRef]
34. Yan, B.; Peng, H.; Fu, J.; Wang, D.; Lu, H. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 10428–10437.
35. Cao, Z.; Huang, Z.; Pan, L.; Zhang, S.; Liu, Z.; Fu, C. Tctrack: Temporal contexts for aerial tracking. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 14778–14788.
36. Kristan, M.; Matas, J.; Leonardis, A. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 2206–2241.

37. Muller, M.; Bibi, A.; Giancola, S.; Alsubaihi, S.; Ghanem, B. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the Computer Vision-ECCV 2018-15th European Conference, Munich, Germany, 8–14 September 2018*; Proceedings, Part I, ser. Lecture Notes in Computer Science; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; Volume 11205, pp. 310–327.
38. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014*; Springer: Cham, Switzerland, 2014; Volume 8693, pp. 740–755.
39. Fan, H.; Lin, L.; Yang, F.; Chu, P.; Deng, G.; Yu, S.; Bai, H.; Xu, Y.; Liao, C.; Ling, H. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 5369–5378.
40. Huang, L.; Zhao, X.; Huang, K. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1562–1577. [CrossRef] [PubMed]
41. Wang, S.; Dai, X.; Ke, C.; Quan, Q. Rflysim: A rapid multicopter development platform for education and research based on Pixhawk and MATLAB. In *Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 15–18 June 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 1587–1594.
42. Cui, Y.; Song, T.; Wu, G.; Wang, L. Mixformerv2: Efficient fully transformer tracking. In *Advances in Neural Information Processing Systems*; Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2023; Volume 36, pp. 58736–58751.
43. Chen, J.; Xu, T.; Huang, B.; Wang, Y.; Li, J. ARTracker: Compute a more accurate and robust correlation filter for uav tracking. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 6514605. [CrossRef]
44. Zuo, H.; Fu, C.; Li, S.; Ye, J.; Zheng, G. Deconnet: End-to-end decontaminated network for vision-based aerial tracking. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5635712. [CrossRef]
45. Chen, X.; Peng, H.; Wang, D.; Lu, H.; Hu, H. Seqtrack: Sequence to sequence learning for visual object tracking. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 14572–14581.
46. Yang, X.; Huang, J.; Liao, Y.; Song, Y.; Zhou, Y.; Yang, J. Light siamese network for long-term onboard aerial tracking. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5623415. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.