PVT++: A Simple End-to-End Latency-Aware Visual Tracking Framework

Bowen Li\textsuperscript{1*}, Ziyuan Huang\textsuperscript{2†}, Junjie Ye\textsuperscript{3}, Yiming Li\textsuperscript{4}, Sebastian Scherer\textsuperscript{1}, Hang Zhao\textsuperscript{5}, Changhong Fu\textsuperscript{3†}
\textsuperscript{1}Carnegie Mellon University, \textsuperscript{2}National University of Singapore, \textsuperscript{3}Tongji University
\textsuperscript{4}New York University, \textsuperscript{5}Tsinghua University
\{bowenli2, basti\}@andrew.cmu.edu, ziyuan.huang@u.nus.edu
\{ye.jun.jie, changhongfu\}@tongji.edu.cn, yimingli@nyu.edu, hangzhao@tsinghua.edu.cn

Abstract

Visual object tracking is essential to intelligent robots. Most existing approaches have ignored the online latency that can cause severe performance degradation during real-world processing. Especially for unmanned aerial vehicles (UAVs), where robust tracking is more challenging and on-board computation is limited, the latency issue can be fatal. In this work, we present a simple framework for end-to-end latency-aware tracking, i.e., end-to-end predictive visual tracking (PVT++). Unlike existing solutions that naively append Kalman Filters after trackers, PVT++ can be jointly optimized, so that it takes not only motion information but also can leverage the rich visual knowledge in most pretrained tracker models for robust prediction. Besides, to bridge the training-evaluation domain gap, we propose a relative motion factor, empowering PVT++ to generalize to the challenging and complex UAV tracking scenes. These careful designs have made the small-capacity lightweight PVT++ a widely effective solution. Additionally, this work presents an extended latency-aware evaluation benchmark for assessing an any-speed tracker in the online setting. Empirical results on a robotic platform from the aerial perspective show that PVT++ can achieve significant performance gain with very little extra latency, obtaining on par or better results than the offline setting.

1. Introduction

Visual object tracking\textsuperscript{1} is fundamental for many robotic applications like navigation [49], cinematography [5], and multi-agent cooperation [9]. Most existing trackers are developed and evaluated under an offline setting [38, 29, 34, 33, 6, 8], where the trackers are assumed to have zero processing time. However, in real-world deployment, the online latency caused by the trackers’ processing time cannot be ignored, since the world would have already changed when the trackers finish processing the captured frame. In particular, with limited onboard computation, this issue is more critical in the challenging unmanned aerial vehicle (UAV) tracking scenes [21, 38, 20]. As shown in Fig. 1, compared with offline setting (gray markers), the latency can cause severe performance degradation during online processing (colored markers). If not handled well, this can easily lead to the failure of robotic applications such as UAV obstacle avoidance [1] and self-localization [61].

To be more specific, the latency hurts online tracking due to: (1) The tracker outputs are always outdated, so there will be mismatch between the tracker result and world state. (2) The trackers can only process the latest frame, so that the non-real-time ones may skip some frames, which makes object motion much larger (see Fig. 2(a) right).

---

\textsuperscript{*}These authors contributed equally to this work.
\textsuperscript{†}Corresponding author
\textsuperscript{1}We focus on single object tracking in this work.
The existence of the latency in real-world applications calls for trackers with prediction capabilities, i.e., predictive trackers. While a standard tracker yields the objects’ location in the input frame (i.e., when it starts processing the input frame, as in Fig. 2(a)), a predictive tracker predicts where the objects could be when it finishes processing the input frame, as illustrated in Fig 2(b).

Existing solutions [32, 36] directly append a Kalman filter (KF) [30] after trackers to estimate the potential object’s location based on its motion model (see Fig 2(c)). However, the rich and readily available visual knowledge from trackers is primarily overlooked, including the object’s appearance and the surrounding environments, which can be naturally exploited to predict the objects’ future paths [51].

To this end, we present a simple framework PVT++ for end-to-end predictive visual tracking. Composed of a tracker and a predictor, PVT++ is able to convert most off-the-shelf trackers into effective predictive trackers. Specifically, to avoid extra latency brought by the predictor, we first design a lightweight network architecture, consisting of a feature encoder, temporal interaction module, and predictive decoder, that leverage both historical motion information and visual cues. By virtue of joint optimization, such a small-capacity network can directly learn from the visual representation provided by most pre-trained trackers for an efficient and accurate motion prediction, as in Fig. 2(c). However, learning this framework is non-trivial due to the training-evaluation domain gap in terms of motion scales. To solve this, we develop a relative motion factor as training objective, so that our framework is independent of the motion scales in training data and can generalize well to the challenging aerial tracking scenes. The integration of lightweight structure and training strategy yields an effective, efficient, and versatile solution.

Beyond methodology, we found that the existing latency-aware evaluation benchmark (LAE) [32] is unable to provide an effective latency-aware comparison for real-time trackers, since it evaluates the result for each frame as soon as it is given. In this case, the latency for any real-time trackers is one frame. Hence, we present an extended latency-aware evaluation benchmark (e-LAE) for any-speed trackers. Evaluated with various latency thresholds, real-time trackers with different speeds can be distinguished.

Empirically, we provide a more general, comprehensive, and practical aerial tracking evaluation for state-of-the-art trackers using our new e-LAE. Converting them into predictive trackers, PVT++ achieves up to 60% improvement under the online setting. As shown in Fig. 1, powered by PVT++, the predictive trackers can achieve comparable or better results than the offline setting. Extensive experiments on multiple tracking models [33, 57, 22] and datasets [47, 16, 37] show that PVT++ works generally for latency-aware tracking, which, to the best of our knowledge, is also the first end-to-end framework for online visual tracking.

## 2. Related Work

### 2.1. Visual Tracking and its Aerial Applications

Visual trackers basically fall into two paradigms, respectively based on discriminative correlation filters [4, 27, 15, 13] and Siamese networks [2, 34, 64, 33, 23, 58]. Compared with general scenarios, aerial tracking is more challenging due to large motions and limited onboard computation resources. Hence, for efficiency, early approaches focus on correlation filters [38, 29, 39, 35]. Later, the development of onboard computation platforms facilitates more robust and applicable Siamese network-based approaches [20, 7, 6, 8]. Most of them are designed under offline settings, ignoring the online latency onboard UAVs, which can lead to severe accuracy degradation. We aim to solve the more
Figure 3. (a) Framework overview of PVT++ for a non-real-time tracker. The tracker has processed frame 0, 2, 5, 8 and obtained corresponding motions \( m \) and visual features \( x, z \). The predictor needs to predict future box \( b_{11}, b_{12} \) based on tracker result \( r_h \). (b) Comparison between LAE (\( \phi(f) \)) [32] and our e-LAE (\( \phi_e(f) \)). For real-time trackers, the mismatch between output and input frames will always be one in LAE (\( \phi(f) - f = 1 \)) regardless of the trackers’ various latency. Differently, e-LAE introduces permitted latency thresholds \( \sigma \in [0, 1] \), which effectively distinguishes the latency difference of distinct models.

2.4. Visual Tracking Benchmarks

Various benchmarks are built for large-scale tracking evaluation [18, 48, 28, 17, 43, 47, 31, 62] with different challenges such as first-person perspective [17], aerial scenes [47], illumination conditions [31, 62], and thermal infrared inputs [43]. Since they all adopt offline evaluation, the influence of the trackers’ latency is ignored. A recent benchmark targets online evaluation [32], but it falls short in real-time trackers and we aim to improve it in this work.

3. Preliminary

We first introduce the latency-aware tracking task here. The input is an image sequence broadcasting with a certain framerate \( \kappa \), denoted as \( (T_f, t_f^W) \), \( f \in \{0, 1, 2, \cdots \} \), where \( t_f^W = \frac{f}{\kappa} \) is the world timestamp and \( f \) is the frame index. Provided with the ground truth box \( B_0 = [x_0, y_0, w_0, h_0] \) at initial 0-th frame, the tracker estimates the boxes in the following frames. Detailed notation table see Appendix C.

Inference. During inference, the tracker finds the latest frame to process when finishing the previous one. Due to the latency, for the \( j \)-th frame that the tracker processes, its index \( j \) may differ from its frame index \( f_j \) in the image sequence. The frame to be processed (frame \( f_j \)) is determined by the tracker timestamp \( t_f^T \) when the model finishes frame \( f_{j-1} \) as follows:

\[
  f_j = \begin{cases} 
  0 & \text{argmax}_{0 \leq f < T} t_f^W \leq t_f^T, \quad j = 0 \\
  \text{others} & \text{otherwise} 
  \end{cases}
\]

With the frame index \( f_j \), the tracker processes frame \( T_{f_j} \) to obtain the corresponding box \( r_{f_j} = [x_{f_j}, y_{f_j}, w_{f_j}, h_{f_j}] \), forming the raw result of the tracker on the frame \( (r_{f_j}, t_{f_j}^T) \). Since tracker may be non-real-time, input frame ids \( f_j, j \in \{0, 1, 2, \cdots \} \) may not be consecutive numbers. For example, in Fig. 3 (a), considering a non-real-time tracker, the processed frames are \( f_j = 0, 2, 5, 8, \cdots \).
Evaluation. Latency-aware evaluation (LAE) [32] compares the ground-truth $b_f$ in frame $I_f$ with the latest result $\hat{b}_f$ from the tracker at $t^W_f$ for evaluation. For standard trackers, the latest result $\hat{b}_f$ to be compared with the ground-truth is obtained as $\hat{b}_f = r_{\phi(f)}$, where $\phi(f)$ is defined as:

$$\phi(f) = \begin{cases} \arg \max_{t_f} t_f^1 \leq t^W_f, & t_f^1 < t^T_0, \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

For instance, in Fig. 3 (b), LAE compares the ground truth $b_3$ with the raw tracker result $r_2$.

4. Extended Latency-Aware Benchmark

Existing latency-aware evaluation [36, 32] adopt Eq. (2) to match the raw output $(r_{f,}, t^T_f)$ to every input frame $f$. However, such a policy fails to reflect the latency difference among real-time trackers. As shown in Fig. 3, since the real-time methods is faster than frame rate, every frame will be processed, i.e., $[f_0, f_1, f_2, \ldots] = [0, 1, 2, \ldots]$. In this case, the latest results will always be from the previous one frame, i.e., using Eq. (2), $\phi(f) = f - 1$. Differently, we extend Eq. (2) to:

$$\phi(f) = \begin{cases} \arg \max_{t_f} t_f^1 \leq t^W_f + \sigma, & t_f^1 < t^T_0, \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $\sigma \in [0, 1)$ is the variable permitted latency. Under e-LAE, $\phi(f)_e$ can be $f - 1$ or $f$ for real-time trackers depending on $\sigma$. For instance, $\phi(f)_e$ would turn from $f - 1$ to $f$ at larger $\sigma$ for slower real-time trackers. This extension distinguishes different real-time trackers (see Section 6.2).

5. Predictive Visual Tracking

Because of the unavoidable latency introduced by the processing time, there is always a mismatch between $\phi(f)$ (or $\phi(f)_e$) and $f$ (when $\sigma$ is small), where $\phi(f)$ is always smaller than $f$, i.e., $\phi(f) < f, f > 0$. To compensate for the mismatch, we resort to predictive trackers that predicts possible location of the object in frame $f$. For the evaluation of $f$-th frame, prior attempts [36, 32] adopt traditional KF [30] to predict the result based on the raw tracking result $r_{\phi(f)}$ in $I_f(f)$ [36], i.e., $\hat{b}_f = \text{KF}(r_{\phi(f)})$. Since previous work [36, 32] are not learnable, neither existing large-scale datasets nor the visual feature are leveraged. Differently, our predictive visual tracking framework PVT++ aims for an end-to-end predictive tracker, which takes both the historical motion and visual features for a more robust and accurate prediction. Note that we use $\hat{b}$ to represent the prediction (results for evaluation) and others are from the tracker output or ground-truth in the following subsections.

5.1. General Framework

As in Fig. 3 (a), PVT++ consists of a tracker $T$ and a predictor $P$. For the $f$-th frame at world time $t^W_f$, the latest result from the tracker is $r_{\phi(f)}$ obtained from frame $I_{\phi(f)}$, i.e., $r_{\phi(f)} = T(x_{\phi(f)}, z)$, where $x_{\phi(f)}$ is the search feature from $I_{\phi(f)}$ and $z$ is the template feature.

After this, the predictor $P$ takes input from the information generated during tracking of the $k$ past frames (including $I_{\phi(f)}$), denoted as Input$_{\phi(f)}$, and predict the position offset normalized by object’s scale, i.e., motion $\tilde{\mathbf{m}}_f = \{\Delta \phi(f) w_{\phi(f)}(\tilde{\mathbf{h}}_{\phi(f)}, \log(\tilde{\mathbf{w}}_{\phi(f)}), \log(\mathbf{h}_{\phi(f)}))\}$, where $\Delta \phi(f)$ and $\Delta \phi(f)$ denote the predicted box center distance between the $f$-th and $\phi(f)$-th frame. $w_{\phi(f)}$ and $h_{\phi(f)}$ are the tracker’s output box scale in frame $\phi(f)$ and $\tilde{\mathbf{w}}_{\phi(f)}$ and $\tilde{\mathbf{h}}_{\phi(f)}$ are the predicted scale in $f$-th frame. With the raw output $r_{\phi(f)}$ at $\phi(f)$ and the motion $\tilde{\mathbf{m}}_f$ from $I_{\phi(f)}$ to the $f$-th frame, the predicted box $\hat{b}_f$ can be easily calculated.

Relative Motion Factor: Due to the large domain gap between the training [52] and evaluation [37] in terms of the absolute motion scale, we find directly using the absolute motion value $\tilde{\mathbf{m}}_f$ as the objective can result in poor performance (see Table 4). Therefore, we define the output of
predictor $P$ to be the relative motion factor based on the average moving speed $p_f$ from the past $k$ frames, which we find is easier to generalize after training:

\[
m_f = P(\text{Input}, \Delta_f) \odot p_f, \quad p_f = \frac{1}{k} \sum_{i=k-b+1}^{j} m_f, \quad \Delta_f = f - \phi(f)
\]

(4)

where $\Delta_f = f - \phi(f)$ denotes the frame interval between current and target frame, and $f$ is the latest processed frame, e.g., $\phi(f)$. $\Delta_f = f_i - f_{i-1}$ denotes the frame interval between $(i-1)$ and $i$-th processed frame. $\odot$ indicates element-wise multiplication.

$m_f$ is the normalized input motion defined as $m_f = \frac{\Delta_f(f_i)}{w_{f_i-1}}, \log(\frac{w_{f_i}}{w_{f_i-1}}), \log(\frac{h_{f_i}}{h_{f_i-1}})$, where $\Delta_f(f_i) = x_{f_i} - x_{f_{i-1}}$ and $\Delta_f(h_i) = y_{f_i} - y_{f_{i-1}}$ are the distance from tracker results $r_f$ and $r_{f-1}$. Such design has made PVT++ agnostic to the specific motion of a dataset, which is crucial for its generalization capability.

We next present the predictor of PVT++ step by step as motion-based $P_m$, visual-appearance-based $P_V$ and multimodal-based $P_M$. All the predictors share the same training objective (Eq. (4)) and a similar structure, consisting of feature encoding, temporal interaction, and predictive decoding as in Fig. 4. In practice, a predictor may need to predict $N$ results, depending on the tracker’s latency.

5.2. Motion-based Predictor

The motion-based predictor $P_m$ only relies on the past motion, i.e., $\text{Input}(f) = m_{f-j+k+1:f_j}$,

\[
\hat{m}_{f,M} = P_m(m_{f-j+k+1:f_j}, \Delta_f) \odot p_f
\]

(5)

where $m_{f-j+k+1:f_j} = [m_{f-j+k+1}, \ldots, m_f] \in \mathbb{R}^{k \times 4}$.

The detailed model structure of the motion predictor $P_m$ is presented in Fig. 4(a). For pre-processing, the motion data $m_{f-j+k+1}, \ldots, m_f$ are first concatenated. Then we apply a fully connected (FC) layer without non-linearity for feature encoding and a 1D convolution followed by activation and global average pooling to obtain the temporally interacted motion feature. In the predictive decoding head, a share FC layer with non-linearity is used for feature mapping. $N$ independent FCs map the feature to $N$ future latent spaces. Finally, the latency features are stacked and transformed to 4 dimension output using a shared FC.

For training, we adopt $L_1$ loss between prediction and ground-truth $L_m = L_1(m_{f,M}, m_f)$.

5.3. Visual Appearance-based Predictor

For efficiency, our visual predictor $P_V$ takes search and template features directly from the tracker backbone as input. Besides, we also find the strong representation in the pre-trained tracker models can boost the small-capacity predictor network. Specifically, template feature $z \in \mathbb{R}^{1 \times C_V \times a \times a}$ is extracted from the given object template patch in the initial frame and search feature $x_{f_j} \in \mathbb{R}^{1 \times C_s \times x \times x}$ is obtained from the $f_j$-th frame patch cropped around $(x_{f_{j-1}}, y_{f_{j-1}})$. Given $k$ past search features $x_{f_j-k+1:f_j} \in \mathbb{R}^{k \times C_s \times x \times x}$ and $z$, we have:

\[
m_{f,V} = P_V(x_{f_j-k+1:f_j}, z, \Delta_f) \odot p_f.
\]

(6)

The detailed model structure of $P_V$ is shown in Fig. 4(b). Inspired by Siamese trackers [33], the feature encoding
stage adopts 1 × 1 convolution before depth-wise correlation (DW-Corr) to produce the similarity map \( x_{f_{j-k+1}:f_{j}} \in \mathbb{R}^{k \times C \times s' \times s'} \). For temporal interaction, we apply 3D convolution and global average pooling.

We find directly training \( P_{V} \) meets convergence difficulty (See Section 6.3). We hypothesize this is because the intermediate similarity map \( x_{f_{j-k+1}:f_{j}} \) fails to provide explicit motion information. To solve this, we introduce an auxiliary branch \( A \), which takes \( x_{f_{j-k+1}:f_{j}} \) as input to obtain the corresponding motion \( m_{f_{j-k+1}:f_{j}}^{s} \),

\[
m_{f_{j-k+1}:f_{j}}^{s} = A(x_{f_{j-k+1}:f_{j}}).
\]

During training, we supervise both the auxiliary branch and the predictive decoder, i.e., \( L_{c} = L_{1}(\hat{m}_{f_{j},V}, m_{f_{j}}) + L_{1}(m_{f_{j-k+1}:f_{j}}; m_{f_{j-k+1}:f_{j}}) \).

5.4. Multi-Modality-based Predictor

The final predictor \( P_{MV} \) is constructed as a combination of motion \( P_{M} \) and visual predictors \( P_{V} \) as,

\[
\hat{m}_{f_{j},MV} = P_{MV}(m_{f_{j-k+1}:f_{j}}, x_{f_{j-k+1}:f_{j}}; z, \Delta f) \odot p_{f_{j}}.
\]

As shown in Fig. 4, the encoding and temporal interaction parts of \( P_{M} \) and \( P_{V} \) run in parallel to form the first two stages of \( P_{MV} \). We concatenate the encoded feature vectors to obtain the multi-modal feature. The predictive decoder follows the same structure to obtain future motions \( \hat{m}_{f_{j},MV} \).

We also tried different fusion strategy in Appendix H.

For training, we add the two additional predictive decoders respectively after motion and visual predictors to help them predict \( \hat{m}_{f_{j},M} \) and \( \hat{m}_{f_{j},V} \), which yields the loss \( L_{MV} = \alpha_{M} L_{M} + \alpha_{V} L_{V} + L_{1}(\hat{m}_{f_{j},MV}, m_{f_{j}}) \). During inference, we only use the joint predictive decoder.

**Remark 1:** The predictors \( P_{M}, P_{V} \) and \( P_{MV} \) can be jointly optimized with tracker \( T \).

6. Experiments

6.1. Implementation Details

**Platform and Datasets.** PVT++ is trained on VID [52], LaSOT [18], and GOT10k [28] using one Nvidia A10 GPU. The evaluation takes authoritative UAV tracking datasets, UAV123, UAV20L [47], DTB70 [37], and UAVDT [16] on typical UAV computing platform, Nvidia Jetson AGX Xavier, for realistic robotic performance. Since the online latency can fluctuate, we run three times and report the average performance. For simplicity, we only consider the tracker’s processing latency during evaluation.

**Metrics.** Following [21], we use two basic metrics, the distance precision (DP) based on center location error (CLE) and area under curve (AUC) based on intersection over union. Under e-LAE, different permitted latency \( \sigma \) corresponds to different DP and AUC, i.e., \( DP_{\sigma} \) and \( AUC_{\sigma} \). We use mDP and mAUC to indicate the area under curve for \( DP_{\sigma} \) and \( AUC_{\sigma} \), \( \sigma \in \{0 : 0.02 : 1\} \).

**Parameters.** For e-LAE, all the evaluated trackers use their official parameters for fairness. To represent the most common case, the image frame rate is fixed to \( \kappa = 30 \) frames/s (FPS) in all the online evaluation. For the PVT++ models, we use \( k = 3 \) past frames. To determine \( N \) for different models, we pre-run the trackers 3 times and record the maximum number of skipped frames, so that when the latency of one specific frame fluctuates, PVT++ can always cover the skipped frame and make sufficient predictions. Detailed training configurations can be found in Appendix B.

6.2. Extended Latency-Aware Evaluation

We evaluate a total of 17 SOTA trackers\(^{2}\) under e-LAE: SiamRPN [34], SiamRPN++ [33], SiamRPN++ [33], SiamMask [57], SiameseFC++ [58], DaSiamRPN [64], and some other baseline approaches.

**Table 1.** The effect of PVT++ on the four SOTA trackers with different inference speeds and backbones. Our models work generally for different cases and can achieve up to 60% performance gain. The best scores are marked out in gray for clear reference. We present some qualitative visualization in Appendix D and the supplementary video.
Table 2. Attribute-based analysis of PVT++ in UAVDT [16]. We found different modality has its specific advantage. Together, the joint model can utilize both and is the most robust under complex UAV tracking challenges. Gray denotes best results.

| Tracker               | Metric | BC   | CR   | SO   | IV   | OB   | SV   | LO   |
|-----------------------|--------|------|------|------|------|------|------|------|
| SiamRPN+++M           | AUC@0.5| 0.764| 0.814| 0.373| 0.503| 0.567| 0.807| 0.540|
|                        | Cr@0.5 | 0.540| 0.523| 0.363| 0.493| 0.554| 0.784| 0.521|
|                        | Cr@0.5 | 0.504| 0.520| 0.538| 0.525| 0.586| 0.584| 0.426|
|                        | Cr@0.5 | 0.505| 0.535| 0.549| 0.545| 0.589| 0.586| 0.511|
| SiamMask              | AUC@0.5| 0.732| 0.549| 0.545| 0.363| 0.493| 0.554| 0.784|
|                        | Cr@0.5 | 0.465| 0.503| 0.491| 0.536| 0.558| 0.542| 0.527|
|                        | Cr@0.5 | 0.488| 0.498| 0.504| 0.495| 0.563| 0.527| 0.541|
|                        | Cr@0.5 | 0.520| 0.522| 0.541| 0.540| 0.596| 0.566| 0.520|

Table 3. Dimension analysis of different modules in PVT++ on DTB70 [37] and UAVDT [16]. Enc_M and Enc_V represent the motion and visual encoders, respectively. Dec_M denotes the joint decoder. ∗ indicates our default setting. We find the channel dimension of PVT++ can be small, so that it introduces very few extra latency on robotics platforms.

| Dim. of Modules | Enc_M | Enc_V | Dec_M | DTB70 | UAVDT |
|-----------------|-------|-------|-------|-------|-------|
| 32              | 0.359 | 0.483 | 0.375 | 0.81  | 0.81  |
| 64              | 0.390 | 0.536 | 0.576 | 0.807 | 0.807 |
| 128             | 0.377 | 0.504 | 0.571 | 0.803 | 0.803 |
| 64              | 0.363 | 0.493 | 0.554 | 0.784 | 0.784 |
| 128             | 0.346 | 0.486 | 0.558 | 0.788 | 0.788 |
| 64              | 0.399 | 0.536 | 0.576 | 0.807 | 0.807 |
| 128             | 0.373 | 0.503 | 0.567 | 0.807 | 0.807 |

SiamAPN [20], SiamAPN++ [7], HiFT [6], SiamGAT [22], SiamBAN [10], SiamCAR [23], ATOM [12], DiMP_S0 [3], DiMP [3], PrDiMP [14], and TrDiMP [56].

As in Fig. 5, we draw curve plots to reflect their performance in AUC and DP metrics under different permitted latency $\sigma$. We report the online mAUC and mDP, offline AUC and DP in the legend. Some offline highly accurate trackers like SiamRPN++M [33], SiamCAR [23], SiamBAN [10], and ATOM [12] can degrade by up to 70% in our online evaluation setting.

Remark 2: e-LAE can better assess the real-time trackers. In DTB70, SiamAPN++ and HiFT are two real-time trackers with HiFT more accurate in success. While since SiamAPN++ is faster, its e-LAE performance will be better.

6.3. Empirical Analysis of PVT++

Overall Effect. To evaluate PVT++, we construct predictive trackers with four well-known methods, i.e., SiamRPN++M [33], SiamRPN++F [33], SiamMask [57], and SiamGAT [22]. As in Table 1, with PVT++, their online performance can be significantly boosted by up to 60%, which sometimes is better than their offline performance. PVT++ also works for recent transformer-based trackers [60, 11], the results can be found in Appendix F.

Remark 3: Real-time trackers [20, 7, 6, 34] perform generally better than non-real-time ones in online evaluation. While we observe that non-real-time trackers empowered by PVT++ can notably outperform real-time ones. E.g., SiamRPN++M [33] with $P_M$ achieves an amazing 0.807 mDP in UAVDT, better than SiamFC++ [58] (0.761).

Table 4. Ablation studies on DTB70 [37]. Official version of PVT++ is marked out in Blackbody. The subscripts $^*$ means predicting raw value instead of motion factor, $^\dagger$ denotes training without auxiliary supervision, and $^\ddagger$ indicates training with tracker fixed. Red denotes improvement and blue represents dropping.

| Ablation Module | Motion Factor$^*$ | Auxiliary Supervision$^\dagger$ | Joint Training$^\ddagger$ |
|-----------------|-------------------|-----------------------------|-------------------------|
| Method Base     | AUC@0.5 | Cr@0.5 | P_M | P_V | $P_{1M}$ | $P_{1V}$ | $P_{1MV}$ | $P_{1MV}^*$ |
| Delta% 0.00     | 0.385   | 0.350  | 0.302 | 0.311 | 0.278  | 0.399  | 0.323  | 0.294  |
| Delta% 0.00     | 0.262   | 0.140  | 0.194 | 0.200 | 0.810  | 0.304  | 0.304  | 0.304  |
| DP@0.5 | 0.387  | 0.350   | 0.302 | 0.311 | 0.278  | 0.399  | 0.323  | 0.294  |
| Delta% 0.00     | 0.351   | 0.120  | 0.194 | 0.200 | 0.810  | 0.304  | 0.304  | 0.304  |

Attribute-based Analysis. For a comprehensive evaluation, we follow [16] and evaluate PVT++ on various challenge attributes$^\dagger$. From Table 2, We found that motion and vision have advantages in different attributes. $P_M$ improves CR and OR, while $P_M$ is good at SO and LO. The joint model $P_{MV}$ makes use of both and is the most robust under various complex aerial tracking challenges. For the full attribute analysis, please see Appendix I.

Dimension Analysis. In addition to its promising performance, PVT++ can also work with very small capacity, which contributes to its lightweight architecture and high efficiency on low-powered UAVs. We analyse the modules of PVT++ with different feature channels in Table 3, where 64 channels for encoders (Enc_M, Enc_V) and 32 channels for the joint decoder (Dec_M) work best. We present more efficiency and complexity comparisons with other motion predictors [63, 24, 40] in Appendix G.

Ablation Studies. We ablate the effect of motion factor prediction, auxiliary branch, and the joint training of PVT++ on DTB70 [37] with SiamRPN++M+ in Table 4. Compared with directly predicting the motion value ($P_{MV}$), using motion factor as the prediction target ($P_M$) can yield much better performance. Removing auxiliary branch $A$ in $P_V$ and $P_{MV}$ to be $P_V^1$ and $P_{MV}^1$, we observe a significant performance drop due to the difficulty in convergence. Joint training the tracker and the predictor ($P_V$ & $P_{MV}$) perform much better than fixing the tracker ($P_V^1$ & $P_{MV}^1$). Training loss of the ablation studies are visualized in Appendix J.

Background cluster (BC), camera rotation (CR), object rotation (OR), small object (SO), illumination variation (IV), object blur (OB), scale variation (SV), and large occlusion (LO).
### Table 5. Averaged results comparison on four datasets [47, 16, 37]. $\mathcal{P}_M$ can achieve better results than prior KF-based solutions [36, 32]. Further introducing visual cues, PVT++ can acquire higher robustness, KF† and † denotes learnable baselines [50, 25].

| Type          | Tracker | Learning-Based | Baseline + $\mathcal{P}_M$ | Baseline | Baseline + KF | Baseline + $\mathcal{P}_M$ | Baseline + $\mathcal{P}_M$ |
|---------------|---------|----------------|-----------------------------|----------|---------------|-----------------------------|-----------------------------|
| Tracker       |         |                |                             |          |               |                             |                             |
|               | AUC@La0 |                |                             |          |               |                             |                             |
| SiamRPN+3K    | 0.43    | 0.402          | 0.473                       | 0.466    | 0.483         | 0.483                       | 0.472                       | **0.485**                  |
| SiamRPN++     | 0.287   | 0.361          | 0.374                       | 0.376    | 0.386         | 0.374                       | 0.345                       | **0.388**                  |
| SiamMask      | 0.557   | 0.607          | 0.639                       | 0.631    | 0.638         | 0.655                       | 0.622                       | **0.663**                  |
| SiamRPN+++M   | 0.423   | 0.502          | 0.523                       | 0.527    | 0.532         | 0.532                       | 0.499                       | **0.542**                  |

Figure 6. Prediction comparison on UAVDT [16]. Red denotes the original trackers, green indicates KF [36] predictions, and blue represents PVT++ prediction. Compared to KF, PVT++ is better at handling in-plane rotation, scale variation, and view point change.

### 6.4. Comparison with KF-based Solutions

#### Quantitative Results.
Prior attempts to latency-aware perception [36, 32] have introduced model-based approach, i.e., KF [30], as predictors. Based on traditional KF, we also designed stronger learnable baselines, KF† [50] and KF§ [25], which adopt the same training as PVT++ models. Basically, KF† [50] learns the two noise matrix and KF§ denotes joint training of KF† and trackers via back-propagation [25]. We compare these KF-based solutions with PVT++ in Table 5, where the same base tracker models are adopted. Compared with the KFs [36, 32], our learning framework holds the obvious advantage in complex UAV tracking scenes. We also observed that PVT++ is very efficient and introduces very little extra latency on the trackers. For specific results per dataset, please refer to Appendix E.

#### Qualitative Results.
To better present the priority of PVT++, some representative scenes are displayed in Fig. 6. Given the same history trajectory, PVT++ holds its advantage against KF-based solution [36]. Especially, when UAV tracking challenges like in-plane rotation (sequence S0103) and aspect ratio change (sequence S0104) appear, PVT++ is capable of fully utilizing the appearance change for robust prediction while simple motion-based KF easily fails.

**Remark 4:** PVT++ outputs $N$ results in a single forward pass instead of autoregressively like KF, resulting in its high efficiency (especially for the motion predictor).

Apart from the robustness priority, PVT++ is also easier to be deployed. Once trained, no further tuning is needed for PVT++ to fit various scenes. Differently, the noise matrix of the KFs is dependent on the environment, which is hard to tune and may not generalize well.

### 6.5. Real-World Tests
We further deploy SiamMask [57] (~1FPS) and SiamRPN++M [33] (~15FPS) with PVT++ on a UAV with Nvidia Jetson AGX Xavier as onboard processor. The onboard speed and center location error (CLE) results are shown in Fig. 7. Despite that the original tracker is not real-time, our PVT++ framework can convert it into a predictive tracker and achieve a good result (CLE < 20 pixels) in real-world tracking. More tests see Appendix M and the video.

### 7. Conclusion
In this work, we present a simple end-to-end framework for latency-aware visual tracking, PVT++, which largely compensates for onboard latency. PVT++ integrates a lightweight predictor module that discovers the visual representation from pre-trained trackers for precise and robust future state estimation. To bridge the training-evaluation domain gap, we propose the relative motion factor, which yields a generalizable framework. In addition to PVT++, we introduce extended latency-aware evaluation benchmark (e-LAE), which assesses an any-speed tracker in the online setting. Extensive evaluations on robotics platform from the challenging aerial perspective show the effectiveness of our PVT++, which improves the offline tracker by up to 60% in the online setting. Real-world tests are further conducted to exhibit the efficacy of PVT++ on physical robots.

### Acknowledgments
This work is conducted when Bowen Li and Ziyuan Huang were visitors at Shanghai Qizhi Institute. This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).
References

[1] Wilbert G Aguilar, Leandro Álvarez, Santiago Grijalva, and Israel Rojas. Monocular Vision-Based Dynamic Moving Obstacles Detection and Avoidance. In Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA), pages 386–398, 2019.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional Siamese Networks for Object Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), pages 850–865, 2016.

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning Discriminative Model Prediction for Tracking. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 6181–6190, 2019.

[4] David S Bolme, J Ross Beveridge, Bruce A Draper, and Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning Discriminative Model Prediction for Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2544–2550, 2010.

[5] Rojerio Bonatti, Cherie Ho, Wenshan Wang, Sanjiban Choudhury, and Sebastian Scherer. Towards a Robust Aerial Cinematography Platform: Localizing and Tracking Moving Targets in Unstructured Environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 229–236, 2019.

[6] Ziang Cao, Changhong Fu, Junjie Ye, Bowen Li, and Yiming Li. HiFT: Hierarchical Feature Transformer for Aerial Tracking. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 15457–15466, 2021.

[7] Ziang Cao, Changhong Fu, Junjie Ye, Bowen Li, and Yiming Li. Siamese Attentional Aggregation Network for Real-Time UAV Tracking. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3086–3092, 2021.

[8] Ziang Cao, Ziyuan Huang, Liang Pan, Zhiwei Liu, and Changhong Fu. TCTrack: Temporal Contexts for Aerial Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2022.

[9] Yu-Jia Chen, Deng-Kai Chang, and Cheng Zhang. Autonomous Tracking Using a Swarm of UAVs: A Constrained Multi-Agent Reinforcement Learning Approach. IEEE Transactions on Vehicular Technology, 69(11):13702–13717, 2020.

[10] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese Box Adaptive Network for Visual Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 6668–6677, 2020.

[11] Yuatao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-End Tracking with Iterative Mixed Attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 13608–13618, 2022.

[12] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate Tracking by Overlap Maximization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4655–4664, 2019.

[13] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient Convolution Operators for Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 6638–6646, 2017.

[14] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic Regression for Visual Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7181–7190, 2020.

[15] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning Spatially Regularized Correlation Filters for Tracking. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 4310–4318, 2015.

[16] Daiwei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), pages 370–386, 2018.

[17] Matteo Dunnhofer, Antonino Furnari, G. Farinella, and C. Micheloni. Is First Person Vision Challenging for Object Tracking? The TREK-100 Benchmark Dataset. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pages 300–317, 2020.

[18] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibing Lin. LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5369–5378, 2019.

[19] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Tracking by Prediction: A Deep Generative Model for Multi-Person Localisation and Tracking. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1122–1132, 2018.

[20] Changhong Fu, Ziang Cao, Yiming Li, Junjie Ye, and Chen Feng. Siamese Anchor Proposal Network for High-Speed Aerial Tracking. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 510–516, 2021.

[21] Changhong Fu, Bowen Li, Fangqiang Ding, Fuling Lin, and Geng Lu. Correlation Filters for Unmanned Aerial Vehicle-Based Aerial Tracking: A Review and Experimental Evaluation. IEEE Geoscience and Remote Sensing Magazine, 10(1):125–160, 2022.

[22] Dongyan Guo, Yanyan Shao, Ying Cui, Zhenhua Wang, Liyan Zhang, and Chunhua Shen. Graph Attention Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 9543–9552, 2021.

[23] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. SiamCAR: Siamese Fully Convolutional
[58] Jinrong Yang, Songtao Liu, Zeming Li, Xiaoping Li, and Jian Sun. Real-time Object Detection for Streaming Perception. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2022.

[60] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint Feature Learning and Relation Modeling for Tracking: A One-Stream Framework. In Proceedings of the European Conference on Computer Vision (ECCV), pages 341–357, 2022.

[61] Junjie Ye, Changhong Fu, Fuling Lin, Fangqiang Ding, Shan An, and Geng Lu. Multi-Regularized Correlation Filter for UAV Tracking and Self-Localization. IEEE Transactions on Industrial Electronics, 69(6):6004–6014, 2022.

[62] Junjie Ye, Changhong Fu, Guangze Zheng, Danda Pani Paudel, and Guang Chen. Unsupervised Domain Adaptation for Nighttime Aerial Tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8896–8905, 2022.

[63] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. SR-LSTM: State Refinement for LSTM towards Pedestrian Trajectory Prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12085–12094, 2019.

[64] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distributor-aware Siamese Networks for Visual Object Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), 2018.