PVT++: A SIMPLE END-TO-END LATENCY-AWARE VISUAL TRACKING FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual object tracking is an essential capability of intelligent robots. Most existing approaches have ignored the online latency that can cause severe performance degradation during real-world processing. Especially for unmanned aerial vehicle, where robust tracking is more challenging and onboard computation is limited, latency issue could 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++). PVT++ is capable of turning most leading-edge trackers into predictive trackers by appending an online predictor. Unlike existing solutions that use model-based approaches, our framework is learnable, such that it can take not only motion information as input but it can also take advantage of visual cues or a combination of both. Moreover, since PVT++ is end-to-end optimizable, it can further boost the latency-aware tracking performance by joint training. Additionally, this work presents an extended latency-aware evaluation benchmark for assessing an *any-speed* tracker in the online setting. Empirical results on robotic platform from aerial perspective show that the motion-based PVT++ can obtain on par or better performance than existing approaches. Further incorporating visual information and joint training techniques, PVT++ can achieve up to **60%** performance gain on various trackers and exhibit better robustness than prior model-based solution, essentially removing the degradation brought by their latency onboard.

1 INTRODUCTION

Visual object tracking[1] is fundamental for many robotic applications like navigation (Nishida et al., 2018), cinematography (Bonatti et al., 2019), and multi-agent cooperation (Chen et al., 2020a), *etc.* Most existing trackers are developed and evaluated under an offline setting (Li et al., 2020c; Huang et al., 2019b; Li et al., 2018; 2019; Cao et al., 2021b; 2022), where the trackers are assumed to have zero processing time. However, in real-world applications, 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, as in Fig. 1(a). If not handled well, this can lead to severe failure of robotic applications such as obstacle avoidance (Aguilar et al., 2019) and self-localization (Ye et al., 2022) for UAVs.

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. 1(a)), a predictive tracker predicts where the objects could be when it *finishes* processing the input frame, as illustrated in Fig 1(b).

Existing solutions to the latency resort to model-based approaches for predicting objects' future locations (Li et al., 2021b; 2020b). Essentially, they use traditional Kalman filter (KF) (Kalman, 1960) to estimate the potential objects' location based on objects' past locations. However, the rich and readily available visual information is primarily overlooked, including the objects' appearance and the surrounding environments, which can be naturally exploited to predict the objects' possible future paths (Rudenko et al., 2020).

This work presents 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 a predictive

[1] We focus on single object tracking in this work.

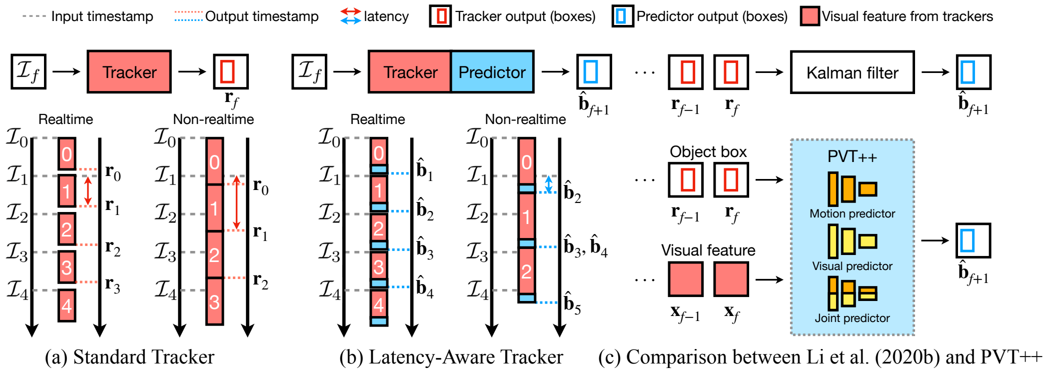

Figure 1: (a) Standard tracker suffers from onboard latency (height of the red boxes). Hence, its result lags behind the world, *i.e.*, $\mathbf{r}_f$ is always obtained after $\mathcal{I}_f$ on the timestamp. (b) Latency-aware trackers introduce predictors to compensate for the latency, which predict the word state, $\hat{\mathbf{b}}_{f+1}$, when finishing the processed frame. (c) Compared with prior KF-based solutions (Li et al., 2020b; 2021b), our end-to-end framework for latency-aware tracking PVT++ leverages both motion and visual feature for prediction.

tracker. Unlike Li et al. (2021b; 2020b) that uses traditional KF (Kalman, 1960) for prediction, our predictor is *end-to-end learnable*. This allows PVT++ to not only leverage historical motion information but also take advantage of the visual features provided by the tracker, as in Fig. 1(c). To this end, we present a simple, effective, efficient, and unified model structure for integrating motion-based and vision-based predictors. Since both the motion and visual features are taken from an existing tracker, the predictor in PVT++ is lightweight, which is crucial for online tracking tasks.

Additionally, we found that the existing latency-aware evaluation benchmark (LAE) (Li et al., 2021b) 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 thresholds for the processing time, real-time trackers with different speeds can now be distinguished by our e-LAE.

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, the motion-based PVT++ obtains comparable or higher performance gain compared with model-based solutions (Li et al., 2020b; 2021b). Further incorporating visual cues and the end-to-end learning strategy, PVT++ achieves up to **60%** improvement under the *online* setting, essentially eliminating the negative effect of onboard latency. Extensive experiments on multiple tracking models and datasets show that PVT++ provides a generic framework for latency-aware tracking, which we hope could facilitate more applicable research in online visual tracking.

2 Related Work

2.1 Visual tracking and its aerial applications

Visual trackers generally fall into two paradigms, respectively based on discriminative correlation filters (Bolme et al., 2010; Henriques et al., 2015; Danelljan et al., 2015; Danelljan et al., 2017) and Siamese networks (Bertinetto et al., 2016; Li et al., 2018; Zhu et al., 2018; Li et al., 2019; Guo et al., 2020; Xu et al., 2020). 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 (Li et al., 2020c; Huang et al., 2019b; Li et al., 2020d;a). Later, the development of onboard computation platforms facilitates more robust and applicable Siamese network-based approaches (Fu et al., 2021; Cao et al., 2021a;b; 2022).

Most of these trackers are designed under offline settings, ignoring the online latency onboard UAVs, which can lead to severe accuracy degradation.

2.2 LATENCY-AWARE PERCEPTION.

Latency of perception systems is first studied in (Li et al., 2020b), which introduces a baseline based on the Kalman-filter (Kalman, 1960) to compensate for the online latency of object detectors. Inspired by this, (Yang et al., 2022) converts a real-time detector into a latency-aware one. More closely related to our work, Li et al. (2021b) present a similar baseline to the solution in (Li et al., 2020b), featuring aerial tracking. Overall, existing works on latency-aware perception adopt only one input modality, *i.e.,* either object's motion (Li et al., 2020b) or visual feature (Yang et al., 2022). In this work, we target aerial tracking and aim to combine both modalities in a unified and end-to-end structure.

2.3 VISUAL TRACKING BENCHMARKS.

Various benchmarks are built for large-scale tracking evaluation (Fan et al., 2019; Müller et al., 2018; Huang et al., 2019a; Dunnhofer et al., 2020; Liu et al., 2020; Mueller et al., 2016; Li et al., 2021a) with different challenges such as first-person perspective (Dunnhofer et al., 2020), aerial scenes (Mueller et al., 2016), illumination conditions (Li et al., 2021a), and thermal infrared inputs (Liu et al., 2020). Since they all adopt *offline* evaluation, the influence of the trackers' latency is ignored. A recent benchmark targets online evaluation (Li et al., 2021b), 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 κ, denoted as $(\mathcal{I}_f, t^{\mathrm{W}}_f), f \in \{0,1,2,\cdots\}$, where $t^{\mathrm{W}}_f = \frac{f}{\kappa}$ is the timestamp of each frame and f is the frame index. Provided with the ground truth box $\mathbf{b}_0 = [x_0, y_0, w_0, h_0]$ at initial 0-th frame, the tracker estimates the boxes in the following frames $\hat{\mathbf{b}}_f, (f > 0)$.

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 time $t^{\mathrm{T}}_{f_{j-1}}$ when the model finishes frame f_{j-1} as follows:

$$f_j = \left\{ \begin{array}{ccc} 0 & , & j=0 \\ \arg\max_f t^{\mathrm{W}}_f \leq t^{\mathrm{T}}_{f_{j-1}} & , & \text{others} \end{array} \right. . \tag{1}$$

With the frame index f_j, the tracker processes frame $\mathcal{I}_{f_j}$ to obtain the corresponding box $\mathbf{r}_{f_j} = [x_{f_j}, y_{f_j}, w_{f_j}, h_{f_j}]$, forming the raw result of the tracker on the frame $(\mathbf{r}_{f_j}, t^{\mathrm{T}}_{f_j})$. 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. 2 (a), considering a non-real-time tracker, the processed frames are $f_j = 0, 2, 4, 8, \cdots$.

Evaluation. Latency-aware evaluation (LAE) (Li et al., 2021b) compares the ground-truth $\mathbf{b}_f$ in frame $\mathcal{I}_f$ with the *latest* result $\hat{\mathbf{b}}_f$ from the tracker at at t^{W}_f for evaluation. For standard trackers, the latest result $\hat{\mathbf{b}}_f$ to be compared with the ground-truth is obtained as $\hat{\mathbf{b}}_f = \mathbf{r}_{\phi(f)}$, where $\phi(f)$ is defined as follows:

$$\phi(f) = \left\{ \begin{array}{ccc} 0 & , & t^{\mathrm{W}}_f < t^{\mathrm{T}}_{p_0} \\ \arg\max_{f_j} t^{\mathrm{T}}_{f_j} \leq t^{\mathrm{W}}_f & , & \text{others} \end{array} \right. . \tag{2}$$

For instance, in Fig. 2 (b), LAE compares the ground truth $\mathbf{b}_3$ with the raw tracker result $\mathbf{r}_2$.

4 EXTENDED LATENCY-AWARE BENCHMARK

Existing latency-aware evaluation (Li et al., 2020b; 2021b) adopt Eq. equation 2 to match the raw output $(\mathbf{r}_{f_j}, t^{\mathrm{T}}_{f_j})$ to every input frame f. However, such a policy fails to reflect the latency difference among real-time trackers. As shown in Fig. 2, since the real-time methods is faster than frame rate, every frame will be processed, *i.e.*, $[f_0, f_1, f_2, \cdots] = [0, 1, 2, \cdots, F]$. In this case, the *latest* results will always be from the previous one frame, *i.e.*, using Eq. equation 2 $\phi(f) = f - 1$. Differently,

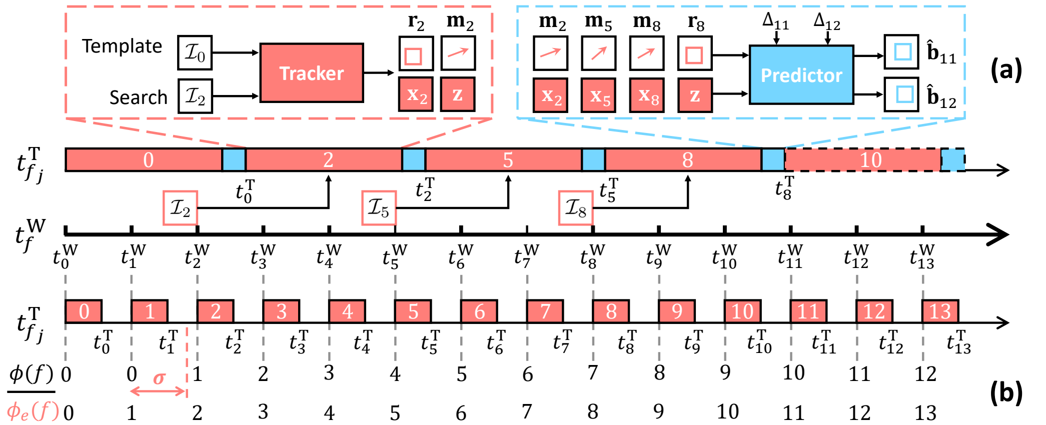

Figure 2: (a) Framework overview of PVT++ for a non-real-time tracker. The tracker has processed frame $0, 2, 5, 8$ and obtained corresponding motions $\mathbf{m}$ and visual features $\mathbf{x}, \mathbf{z}$. The predictor needs to predict future box $\hat{\mathbf{b}}_{11}$, $\hat{\mathbf{b}}_{12}$ based on tracker result $\mathbf{r}_8$. (b) Comparison between LAE ($\phi(f)$) (Li et al., 2021b) and our e-LAE ($\phi_e(f)$). For real-time trackers, the mismatch between output and input frames will always be one in LAE ($\phi(f) - f \equiv 1$) regardless of the tracker latency. Differently, e-LAE introduces permitted latency thresholds $\sigma \in [0, 1)$, which effectively evaluates the latency.

we extend Eq. equation 2 to:

$$\phi(f)_{\mathrm{e}} = \begin{cases} 0 & , \quad t_f^{\mathrm{W}} < t_{f_0}^{\mathrm{T}} \\ \arg\max_{f_j} t_{f_j}^{\mathrm{T}} \leq t_f^{\mathrm{W}} + \sigma & , \quad \text{others} \end{cases} , \tag{3}$$

where $\sigma \in [0, 1)$ is the variable permitted latency. Under e-LAE, $\phi(f)_{\mathrm{e}}$ can be $f - 1$ or f for real-time trackers depending on σ, and $\phi(f)_{\mathrm{e}}$ turns from $f - 1$ to f at different permitted latency σ for real-time trackers with different latency. This extension distinguishes different real-time trackers.

5 Predictive Visual Tracking

Because of the unavoidable latency introduced by the processing time, there is always a mismatch between $\phi(f)$ (or $\phi(f)_{\mathrm{e}}$) and f (when σ 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 (Li et al., 2020b; 2021b) adopt traditional KF (Kalman, 1960) to predict the result based on the raw tracking result $\mathbf{r}_{\phi(f)}$ in $\mathcal{I}_{\phi}(f)$ (Li et al., 2020b), *i.e.,* $\hat{\mathbf{b}}_f = \mathrm{KF}(\mathbf{r}_{\phi(f)})$. Since it is not learnable, it cannot leverage existing large-scale datasets or the visual feature. 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 of $\hat{\mathbf{b}}_f$. Note that we use $\hat{\cdot}$ to represent the prediction and others are from the tracker output or ground-truth in the following subsections.

5.1 General Framework

As in Fig. 2 (a), PVT++ consists of a tracker $\mathcal{T}$ and a predictor $\mathcal{P}$. For the f-th frame at t_f^{W}, the latest result from the tracker is $\mathbf{r}_{\phi(f)}$ obtained from frame $\mathcal{I}_{\phi(f)}$, *i.e.,* $\mathbf{r}_{\phi(f)} = \mathcal{T}(\mathbf{x}_{\phi(f)}, \mathbf{z})$, where $\mathbf{x}_{\phi(f)}$ is the search feature from $\mathcal{I}_{\phi(f)}$ and $\mathbf{z}$ is the template feature.

After this, the predictor $\mathcal{P}$ takes input from the information generated during tracking of the k past frames (including $\mathcal{I}_{\phi(f)}$), denoted as $\mathrm{Input}_{\phi(f)}$, and predict the position offset normalized by object's scale, *i.e.,* motion $\hat{\mathbf{m}}_f$:

$$\hat{\mathbf{m}}_f = [\frac{\Delta_{\hat{x}}(f)}{w_{\phi(f)}}, \frac{\Delta_{\hat{y}}(f)}{h_{\phi(f)}}, \mathrm{Log}(\frac{\hat{w}_f}{w_{\phi(f)}}), \mathrm{Log}(\frac{\hat{h}_f}{h_{\phi(f)}})] = \mathcal{P}\Big(\mathrm{Input}_{\phi(f)}, \Delta_f\Big) , \tag{4}$$

where $\Delta_f = f - \phi(f)$ indicates the frame interval between the latest frame and the f-th frame. $\Delta_{\hat{x}}(f)$ and $\Delta_{\hat{y}}(f)$ denote the predicted box distance between the f-th and $\phi(f)$-th frame. $w_{\phi(f)}$

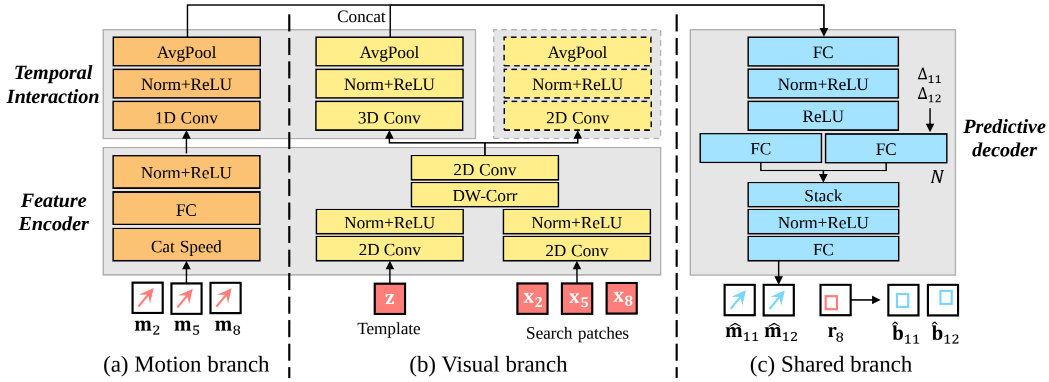

Figure 3: Detailed model structure of the predictor modules in PVT++. The models shares similar architecture, *i.e.*, feature encoder, temporal interaction, and predictive decoder. We present the motion branch, visual branch, and share decoding branch in (a), (b), and (c), respectively. Note that the dashed blocks denote auxiliary branch, which only exists in training. The input and output are in correspondence to the case in Fig. 2 (a).

and $h_{\phi(f)}$ are the tracker's output box scale in frame $\phi(f)$ and $\hat{w}_f$, $\hat{h}_f$ are the predicted scale in f-th frame. Here, the history information $\text{Input}_{\phi(f)}$ can both be object motion and the visual cue from the tracker, depending on the type of predictor. With the raw output $\mathbf{r}_{\phi(f)}$ at $\phi(f)$ and the motion $\hat{\mathbf{m}}_f$ from $\mathcal{I}_{\phi(f)}$ to the f-th frame, the predicted box $\hat{\mathbf{b}}_f$ can be easily calculated.

Due to the large gap between the datasets for training (Russakovsky et al., 2015) and evaluation (Li & Yeung, 2017) in terms of the absolute motion scale, we find directly using the absolute motion value $\hat{\mathbf{m}}_f$ as the objective can result in poor performance. In practice, we predict the relative motion factor based on the average moving speed $\mathbf{p}_{f_j}$ from the past k frames, which is easier to generalize to datasets with various motion scales after training:

$$\hat{\mathbf{m}}_f = \mathcal{P}\big(\text{Input}_{\phi(f)}, \Delta_f\big) \odot \mathbf{p}_{f_j}\ , \quad \mathbf{p}_{f_j} = \sum_{i=1}^{k} \big(\frac{1}{k} \odot \frac{\mathbf{m}_{f_{j-i+1}}}{\Delta_{f_{j-i+1}}}\big), \tag{5}$$

where $\Delta_{f_{j-i+1}} = f_{j-i+1} - f_{j-i}$ denotes the frame interval and $\odot$ is the element-wise multiplication. $\mathbf{m}_{f_j}$ is the normalized input motion defined as follows:

$$\mathbf{m}_{f_j} = [\frac{\Delta_x(f_j)}{w_{f_{j-1}}}, \frac{\Delta_y(f_j)}{h_{f_{j-1}}}, \text{Log}(\frac{w_{f_j}}{w_{f_{j-1}}}), \text{Log}(\frac{h_{f_j}}{h_{f_{j-1}}})]\ , \tag{6}$$

where $\Delta_x(f_j) = x_{f_j} - x_{f_{j-1}}$ and $\Delta_y(f_j) = y_{f_j} - y_{f_{j-1}}$ are the distance from $\mathbf{r}_{f_j}$ and $\mathbf{r}_{f_{j-1}}$.

We introduce three types of predictors, which are motion-based $\mathcal{P}_{\text{M}}$, visual-appearance-based $\mathcal{P}_{\text{V}}$ and multi-modal-based $\mathcal{P}_{\text{MV}}$. All predictors share a similar structure consisting of feature encoding, temporal interaction, and predictive decoding as in Fig. 3. Based on each set of k past frames, a predictor may need to predict the result for multiple frames depending on the tracker's latency. In this case, we share most of the structure except for the second last fully connected layer as in Fig. 3 (c), which we select based on the frame distance Δ_f for prediction during inference.

5.2 Motion-based Predictor

Our motion-based predictor $\mathcal{P}_{\text{M}}$ only relies on the past motion, *i.e.*, $\text{Input}_{\phi(f)} = \mathbf{m}_{f_{j-k+1}:f_j}$,

$$\hat{\mathbf{m}}_{f,\text{M}} = \mathcal{P}_{\text{M}}\big(\mathbf{m}_{f_{j-k+1}:f_j}, \Delta_f\big) \odot \mathbf{p}_{f_j}\ , \tag{7}$$

where $\mathbf{m}_{f_{j-k+1}:f_j} = [\mathbf{m}_{f_{j-k+1}}, \cdots, \mathbf{m}_{f_j}] \in \mathbb{R}^{k\times 4}$.

The detailed model structure of the motion predictor $\mathcal{P}_{\text{M}}$ is presented in Fig. 3(a). For pre-processing, the motion data $\mathbf{m}_{f_{j-k+1}}, \cdots, \mathbf{m}_{f_j}$ are first concatenated. Then we apply a fully connected layer with non-linearity for feature encoding and a 1D convolution followed by activation

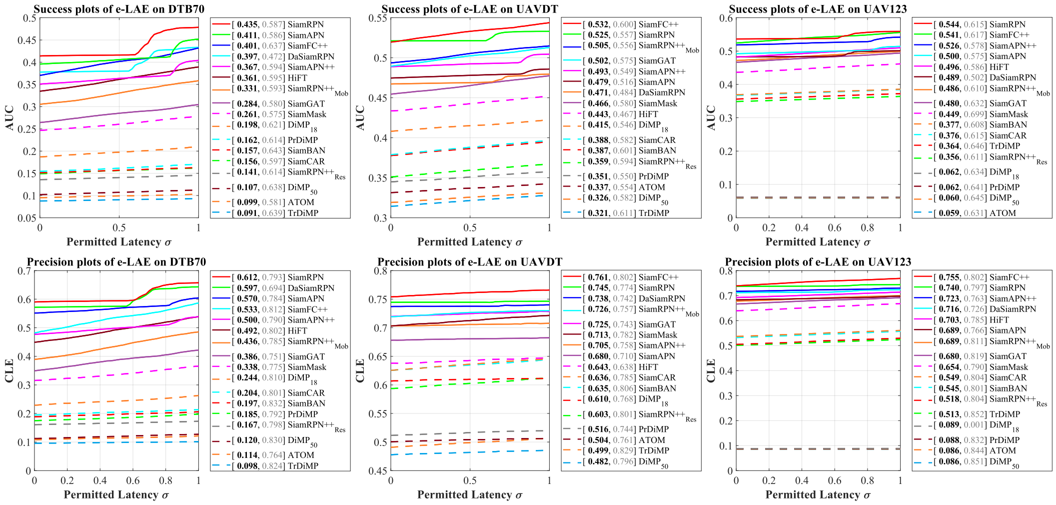

Figure 4: The performance of the SOTA trackers in authoritative UAV tracking benchmarks under our e-LAE benchmark. We report [online mAUC and mDP, offline AUC and DP] in the legend. All trackers struggle to overcome onboard latency in online tracking.

and global average pooling to obtain the temporally interacted motion feature. In the predictive decoding head, a share fully connected layer (FC) 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 $\mathcal{L}_1$ loss between prediction and ground-truth $\mathcal{L}_{\mathrm{M}} = \mathcal{L}_1(\hat{\mathbf{m}}_{f,M}, \mathbf{m}_f)$.

5.3 Visual Appearance-based Predictor

Since our PVT++ is learnable, more information like visual appearance can be leveraged for better prediction. For efficiency, our visual predictor $\mathcal{P}_{\mathrm{V}}$ takes search and template features from the tracker backbone as input. Specifically, template feature $\mathbf{z} \in \mathbb{R}^{1\times C_{\mathrm{V}}\times a\times a}$ is extracted from the given object template patch in the initial frame and search feature $\mathbf{x}_{f_j} \in \mathbb{R}^{1\times C_{\mathrm{V}}\times s\times s}$ is extracted from the patch cropped around the center of the object in the last processed image, frame f_{j-1}. This also enables joint training to make the visual features more ready for prediction. Given k past search features $\mathbf{x}_{f_{j-k+1}:f_j} = [\mathbf{x}_{f_{j-k+1}}, \cdots, \mathbf{x}_{f_j}] \in \mathbb{R}^{k\times C_{\mathrm{V}}\times s\times s}$ and $\mathbf{z}$, our visual predictor can be expressed as:

$$\hat{\mathbf{m}}_{f,\mathrm{V}} = \mathcal{P}_{\mathrm{V}}\big(\mathbf{x}_{f_{j-k+1}:f_j}, \mathbf{z}, \Delta_f\big) \odot \mathbf{p}_{f_j} \ . \tag{8}$$

The detailed model structure of $\mathcal{P}_{\mathrm{V}}$ is shown in Fig. 3(b). Inspired by Siamese trackers (Li et al., 2019), the feature encoding stage adopts 1×1 convolution before depth-wise correlation (DW-Corr) to produce the similarity map $\mathbf{x}^{\mathrm{e}}_{f_{j-k+1}:f_j} \in \mathbb{R}^{k\times C_{\mathrm{V}}\times s'\times s'}$. For temporal interaction, we apply 3D convolution and global average pooling.

We find directly training $\mathcal{P}_{\mathrm{V}}$ meets convergence difficulty. We hypothesize this is because $\mathbf{x}^{\mathrm{e}}_{f_{j-k+1}:f_j}$ doesn't contain explicit motion information and to accelerate convergence, we introduce an auxiliary branch $\mathcal{A}$, which takes $\mathbf{x}^{\mathrm{e}}_{f_{j-k+1}:f_j}$ as input to predict the corresponding motion $\hat{\mathbf{m}}_{f_{j-k+1}:f_j}$,

$$\hat{\mathbf{m}}_{f_{j-k+1}:f_j} = \mathcal{A}(\mathbf{x}^{\mathrm{e}}_{f_{j-k+1}:f_j}). \tag{9}$$

For training, we supervise both the auxiliary branch and the predictive decoder, *i.e.*, $\mathcal{L}_{\mathrm{V}} = \mathcal{L}_1(\hat{\mathbf{m}}_{f,\mathrm{V}}, \mathbf{m}_f) + \mathcal{L}_1(\hat{\mathbf{m}}_{f_{j-k+1}:f_j}, \mathbf{m}_{f_{j-k+1}:f_j})$.

5.4 Multi-modal-based Predictor

The multi-modal predictor is constructed as a combination of motion $\mathcal{P}_{\mathrm{M}}$ and visual predictors $\mathcal{P}_{\mathrm{V}}$, which takes both visual and motion information as input, *i.e.*,

$$\hat{\mathbf{m}}_{f,\mathrm{MV}} = \mathcal{P}_{\mathrm{MV}}\big(\mathbf{m}_{f_{j-k+1}:f_j}, \mathbf{x}_{f_{j-k+1}:f_j}, \mathbf{z}, \Delta_f\big) \odot \mathbf{p}_{f_j} \ . \tag{10}$$

As shown in Fig. 3, the encoding and temporal interaction parts of $\mathcal{P}_{\mathrm{M}}$ and $\mathcal{P}_{\mathrm{V}}$ run in parallel to form the first two stages of $\mathcal{P}_{\mathrm{MV}}$. We concatenate the encoded feature vectors to obtain the multi-modal

Table 1: The effect of PVT++ on the three SOTA trackers with different speeds. Our models work generally 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 supplementary video.

Tracker	Dataset / PVT++	DTB70 AUC@La0$_{\Delta\%}$	DTB70 DP@La0$_{\Delta\%}$	UAVDT AUC@La0$_{\Delta\%}$	UAVDT DP@La0$_{\Delta\%}$	UAV20L AUC@La0$_{\Delta\%}$	UAV20L DP@La0$_{\Delta\%}$	UAV123 AUC@La0$_{\Delta\%}$	UAV123 DP@La0$_{\Delta\%}$
SiamRPN++$_{\mathrm{Mob}}$ (21FPS)	N/A	0.305$_{+0.00}$	0.387$_{+0.00}$	0.494$_{+0.00}$	0.719$_{+0.00}$	0.448$_{+0.00}$	0.619$_{+0.00}$	0.472$_{+0.00}$	0.678$_{+0.00}$
	$\mathcal{P}_{\mathrm{M}}$	0.385$_{+26.2}$	0.523$_{+35.1}$	0.529$_{+7.10}$	0.745$_{+3.60}$	0.481$_{+7.40}$	0.647$_{+4.50}$	0.537$_{+13.8}$	0.737$_{+8.70}$
	$\mathcal{P}_{\mathrm{V}}$	0.352$_{+15.4}$	0.472$_{+22.0}$	0.564$_{+14.2}$	0.799$_{+11.1}$	0.488$_{+8.90}$	0.675$_{+9.00}$	0.504$_{+6.80}$	0.703$_{+3.70}$
	$\mathcal{P}_{\mathrm{MV}}$	0.399$_{+30.8}$	0.536$_{+38.5}$	0.576$_{+16.6}$	0.807$_{+12.2}$	0.508$_{+13.4}$	0.697$_{+12.6}$	0.537$_{+13.8}$	0.741$_{+9.30}$
SiamRPN++$_{\mathrm{Res}}$ (5FPS)	N/A	0.136$_{+0.00}$	0.159$_{+0.00}$	0.351$_{+0.00}$	0.594$_{+0.00}$	0.310$_{+0.00}$	0.434$_{+0.00}$	0.349$_{+0.00}$	0.505$_{+0.00}$
	$\mathcal{P}_{\mathrm{M}}$	0.199$_{+46.3}$	0.258$_{+62.3}$	0.449$_{+27.9}$	0.684$_{+15.2}$	0.404$_{+30.3}$	0.560$_{+29.0}$	0.442$_{+26.6}$	0.627$_{+24.2}$
	$\mathcal{P}_{\mathrm{V}}$	0.179$_{+31.6}$	0.225$_{+41.5}$	0.403$_{+14.8}$	0.665$_{+12.0}$	0.398$_{+28.4}$	0.548$_{+26.3}$	0.398$_{+14.0}$	0.559$_{+10.7}$
	$\mathcal{P}_{\mathrm{MV}}$	0.205$_{+50.7}$	0.256$_{+61.0}$	0.488$_{+39.0}$	0.726$_{+22.2}$	0.416$_{+34.2}$	0.568$_{+30.9}$	0.442$_{+26.6}$	0.619$_{+22.6}$
SiamMask (12FPS)	N/A	0.247$_{+0.00}$	0.313$_{+0.00}$	0.455$_{+0.00}$	0.703$_{+0.00}$	0.405$_{+0.00}$	0.571$_{+0.00}$	0.436$_{+0.00}$	0.639$_{+0.00}$
	$\mathcal{P}_{\mathrm{M}}$	0.370$_{+49.8}$	0.508$_{+62.3}$	0.531$_{+16.7}$	0.760$_{+8.10}$	0.449$_{+10.9}$	0.607$_{+6.30}$	0.532$_{+22.0}$	0.743$_{+16.9}$
	$\mathcal{P}_{\mathrm{V}}$	0.292$_{+18.2}$	0.405$_{+29.4}$	0.532$_{+16.9}$	0.777$_{+10.5}$	0.430$_{+6.20}$	0.601$_{+5.30}$	0.503$_{+15.4}$	0.705$_{+10.3}$
	$\mathcal{P}_{\mathrm{MV}}$	0.342$_{+29.5}$	0.463$_{+47.9}$	0.566$_{+24.4}$	0.797$_{+13.4}$	0.469$_{+15.8}$	0.644$_{+12.8}$	0.536$_{+22.9}$	0.749$_{+17.2}$

Table 2: Attribute-based analysis of PVT++ on SiamRPN++$_{\mathrm{Mob}}$ in UAVDT dataset. We found different modality has their specific advantage. Together, the joint model can utilize both and become the most robust under complex UAV tracking challenges. Gray denotes best results.

Metric	AUC@La0								DP@La0							
Att.	BC	CR	OR	SO	IV	OB	SV	LO	BC	CR	OR	SO	IV	OB	SV	LO
base	0.448	0.45	0.438	0.494	0.539	0.525	0.49	0.422	0.659	0.643	0.638	0.779	0.777	0.772	0.68	0.569
$\mathcal{P}_{\mathrm{M}}$	0.461	0.495	0.481	0.549	0.578	0.542	0.505	0.521	0.666	0.684	0.681	0.815	0.811	0.778	0.691	0.717
$\mathcal{P}_{\mathrm{V}}$	0.504	0.52	0.538	0.525	0.588	0.568	0.584	0.436	0.733	0.72	0.753	0.793	0.835	0.822	0.796	0.585
$\mathcal{P}_{\mathrm{MV}}$	0.505	0.535	0.549	0.545	0.599	0.589	0.586	0.511	0.727	0.732	0.764	0.814	0.848	0.846	0.794	0.694

feature. The predictive decoder follows the same structure to obtain future motions $\hat{\mathbf{m}}_{f,\mathrm{MV}}$. We also tried different fusion strategy in Appendix G.

For training, we add two additional predictive decoders respectively after motion and visual predictors to help them predict $\hat{\mathbf{m}}_{f,\mathrm{M}}$ and $\hat{\mathbf{m}}_{f,\mathrm{V}}$, which yields the loss $\mathcal{L}_{\mathrm{MV}} = \alpha_{\mathrm{M}}\mathcal{L}_{\mathrm{M}} + \alpha_{\mathrm{V}}\mathcal{L}_{\mathrm{V}} + \mathcal{L}_1(\hat{\mathbf{M}}_{f,\mathrm{MV}}, \mathbf{M}_f)$. During inference, we only use the joint predictive decoder.

6 Experiments

6.1 Implementation Details

Platform and datasets. PVT++ is trained on VID (Russakovsky et al., 2015), LaSOT (Fan et al., 2019), and GOT10k (Huang et al., 2019a) using one Nvidia A10 GPU. The evaluation takes authoritative UAV tracking datasets, UAV123, UAV20L (Mueller et al., 2016), DTB70 (Li & Yeung, 2017), and UAVDT (Du et al., 2018) 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.

Metrics. Following (Fu et al., 2022), 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 σ corresponds to different DP and AUC, *i.e.,* DP@Laσ and AUC@Laσ. We use mDP and mAUC to indicate the area under cure for DP@Laσ and AUC@Laσ, $\sigma \in [0 : 0.02 : 1)$.

Parameters. For e-LAE, all the evaluated trackers use their official parameters for fairness. For all PVT++ models, we use $k = 3$ past frames, while N varies for different models with different latency. Detailed training parameters can be referred to the code and Appendix B.

6.2 Extended Latency-aware Evaluation

We evaluate a total of 17 SOTA trackers[2] under e-LAE: SiamRPN (Li et al., 2018), SiamRPN++$_{\mathrm{Mob}}$ (Li et al., 2019), SiamRPN++$_{\mathrm{Res}}$ (Li et al., 2019), SiamMask (Wang et al., 2019), SiameseFC++ (Xu et al., 2020), DaSiamRPN (Zhu et al., 2018), SiamAPN (Fu et al., 2021), SiamAPN++ (Cao et al., 2021a), HiFT (Cao et al., 2021b), SiamGAT (Guo et al., 2021),

[2]Subscripts denote the backbone used, *i.e.,* MobileNet (Sandler et al., 2018), and ResNet 18 or 50 (He et al., 2018).

Table 3: Ablation studies on DTB70 (Li & Yeung, 2017). Official version of PVT++ is marked out in Blackbody. Red denotes improvement and blue represents dropping.

Discrip.		Motion factor		Auxiliary branch				Joint training			
Method	Base	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{M}}^{\dagger}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{V}}^{\dagger}$	$\mathcal{P}_{\mathrm{MV}}$	$\mathcal{P}_{\mathrm{MV}}^{\dagger}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{V}}^{\ddagger}$	$\mathcal{P}_{\mathrm{MV}}$	$\mathcal{P}_{\mathrm{MV}}^{\ddagger}$
AUC@La0	0.305	**0.385**$_{+26.2}$	0.3$_{-1.60}$	**0.352**$_{+15.4}$	0.278$_{-8.90}$	**0.399**$_{+30.8}$	0.294$_{-3.60}$	**0.352**$_{+15.4}$	0.311$_{+2.00}$	**0.399**$_{+30.8}$	0.323$_{+5.90}$
DP@La0	0.387	**0.523**$_{+35.1}$	0.383$_{-1.00}$	**0.472**$_{+22.0}$	0.349$_{-9.80}$	**0.536**$_{+38.5}$	0.387$_{-0.00}$	**0.472**$_{+22.0}$	0.412$_{+6.50}$	**0.536**$_{+38.5}$	0.429$_{-10.9}$

Table 4: Dimension analysis of different modules in PVT++ on DTB70 (Li & Yeung, 2017) and UAVDT (Du et al., 2018). $\mathrm{Enc.}_{\mathrm{M}}$ and $\mathrm{Enc.}_{\mathrm{V}}$ represent the motion and visual encoders, respectively. $\mathrm{Dec.}_{\mathrm{MV}}$ 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.	DTB70		UAVDT		Dim.	DTB70		UAVDT		Dim.	DTB70		UAVDT	
$\mathrm{Enc.}_{\mathrm{M}}$	mAUC	mDP	mAUC	mDP	$\mathrm{Enc.}_{\mathrm{V}}$	mAUC	mDP	mAUC	mDP	$\mathrm{Dec.}_{\mathrm{MV}}$	mAUC	mDP	mAUC	mDP
N/A	0.305	0.387	0.494	0.719	N/A	0.305	0.387	0.494	0.719	N/A	0.305	0.387	0.494	0.719
16	0.357	0.479	0.565	0.797	16	0.362	0.487	0.545	0.772	16	0.369	0.496	0.572	0.804
32	0.359	0.483	0.575	0.81	32	0.363	0.493	0.554	0.784	**32***	**0.399**	**0.536**	**0.576**	**0.807**
64*	**0.399**	**0.536**	**0.576**	**0.807**	**64***	**0.399**	**0.536**	**0.576**	**0.807**	64	0.373	0.503	0.567	0.807
128	0.373	0.504	0.571	0.803	128	0.364	0.486	0.558	0.788	128	0.362	0.485	0.561	0.791

SiamBAN (Chen et al., 2020b), SiamCAR (Guo et al., 2020), ATOM (Danelljan et al., 2019), DiMP_{50} (Bhat et al., 2019), DiMP_{18} (Bhat et al., 2019), PrDiMP (Danelljan et al., 2020), and TrDiMP (Wang et al., 2021).

As in Fig. 4, we draw curve plots to reflect their performance in AUC and DP metrics under different permitted latency σ. We report the [online mAUC and mDP, offline mAUC and mDP] in the legend. Some offline highly accurate trackers like $\mathrm{SiamRPN{+}{+}}_{\mathrm{Res}}$ (Li et al., 2019), SiamCAR (Guo et al., 2020), SiamBAN (Chen et al., 2020b), and ATOM (Danelljan et al., 2019) can degrade by up to **70%** in our online evaluation setting.

Note that e-LAE can better assess the real-time trackers by taking the efficiency into account. In DTB70, SiamAPN++ and HiFT are real-time trackers with HiFT slightly more accurate (in success). However, since SiamAPN++ is much faster, its e-LAE performance is better.

6.3 Empirical analysis

Performance of PVT++. To evaluate PVT++, we construct predictive trackers with three well-known trackers, *i.e.,* $\mathrm{SiamRPN{+}{+}}_{\mathrm{Mob}}$ (Li et al., 2019), $\mathrm{SiamRPN{+}{+}}_{\mathrm{Res}}$ (Li et al., 2019), and SiamMask (Wang et al., 2019). As in Table 1, with PVT++, their online performance can be significantly boosted by up to **60%**. Further, although real-time trackers (Li et al., 2020c; Fu et al., 2022; Li et al., 2021a; Fu et al., 2021; Cao et al., 2021a;b; 2022) perform generally better than non-real-time trackers in online evaluation, we observe that non-real-time trackers empowered by PVT++ can notably outperform real-time ones without PVT++ (*e.g.,* **0.807** mDP of $\mathrm{SiamRPN{+}{+}}_{\mathrm{Mob}}$ with $\mathcal{P}_{\mathrm{MV}}$ in UAVDT *vs.* 0.745 of the real-time tracker SiamRPN).

Attribute-based analysis. For a comprehensive evaluation, we follow (Du et al., 2018) and evaluate PVT++ on various challenge attributes[3]. We found that motion and vision have advantages in different attributes. $\mathcal{P}_{\mathrm{V}}$ improves CR and OR, while $\mathcal{P}_{\mathrm{M}}$ is good at SO and LO. The joint model $\mathcal{P}_{\mathrm{MV}}$ makes use of both and is the most robust under various complex aerial tracking challenges. For the full attribute analysis, please see Appendix H.

Ablation studies. We ablate the effect of motion factor prediction, auxiliary branch, and the joint training of PVT++ on DTB70 (Li & Yeung, 2017) with $\mathrm{SiamRPN{+}{+}}_{\mathrm{Mob}}$ in Table 3. Compared with directly predicting the motion value ($\mathcal{P}_{\mathrm{M}}^{\dagger}$), using *motion factor* as the prediction target ($\mathcal{P}_{\mathrm{M}}$) can yield much better performance. Removing *auxiliary branch* $\mathcal{A}$ in $\mathcal{P}_{\mathrm{V}}$ and $\mathcal{P}_{\mathrm{MV}}$ to be $\mathcal{P}_{\mathrm{V}}^{\dagger}$ and $\mathcal{P}_{\mathrm{MV}}^{\dagger}$, we observe a significant performance drop due to the difficulty in convergence. *Joint training* the tracker and the predictor ($\mathcal{P}_{\mathrm{V}}$ & $\mathcal{P}_{\mathrm{MV}}$) perform much better than fixing the tracker ($\mathcal{P}_{\mathrm{V}}^{\ddagger}$ and $\mathcal{P}_{\mathrm{MV}}^{\ddagger}$). Training loss curves of the ablation studies are further presented in Appendix I.

[3]Background cluter (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: Comparison between our learning based PVT++ and prior KF-based solution (Li et al., 2020b). The motion based PVT++ can achieve on par or better results. Further introducing visual cues, PVT++ can acquire higher robustness. We also designed stronger learnable KF baselines, KF† and ‡, which are still less robust than our $\mathcal{P}_{\mathrm{MV}}$. Best scores are marked out in gray.

Tracker	SiamRPN$_{\mathrm{Mob}}$						SiamRPN$_{\mathrm{Res}}$						SiamMask					
Pred.	KF	KF†	KF‡	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	KF	KF†	KF‡	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	KF	KF†	KF‡	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$
mAUC	0.462	0.466	0.481	0.483	0.477	0.505	0.441	0.458	0.468	0.471	0.439	0.478	0.361	0.376	0.386	0.374	0.345	0.388
mDP	0.639	0.642	0.658	0.663	0.662	0.695	0.607	0.631	0.639	0.655	0.622	0.663	0.502	0.527	0.532	0.532	0.499	0.542

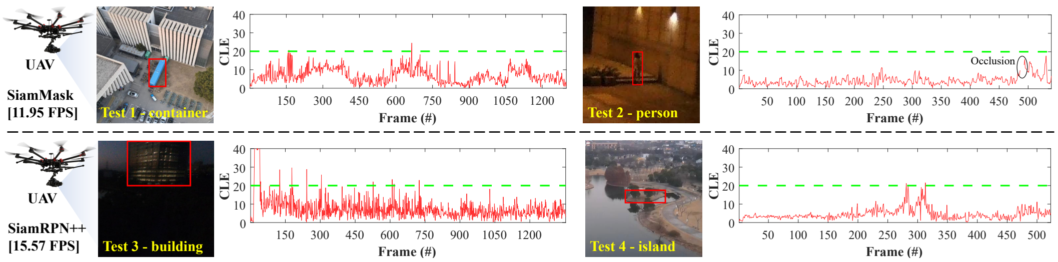

Figure 5: Real-world tests of PVT++. Thanks to PVT++, the non-real-time trackers work effectively under real-world tracking challenges like scale variation in Test 1 and occlusion in Test 2.

Dimension analysis. In addition to its promising performance, PVT++ can also work with small channel dimensions, which contributes to its light weight and efficiency on low-powered UAVs. We analyse the modules of PVT++ with different feature channels in Table 4, where 64 channels for encoders ($\mathrm{Enc.}_{\mathrm{M}}$, $\mathrm{Enc.}_{\mathrm{V}}$) and 32 channels for the joint decoder ($\mathrm{Dec.}_{\mathrm{J}}$) work best.

Comparison with KF. Prior attempts to latency-aware perception (Li et al., 2020b; 2021b) have introduced model-based approach, *i.e.*, KF (Kalman, 1960), as predictors. Based on traditional KF, we also designed stronger learnable baselines, KF† and KF‡, which adopt the same training as PVT++. Basically, KF† learns the two noise matrix and KF‡ denotes joint training of KF† and trackers. We compare such solutions with our PVT++ in Table 5, where the same base tracker models are adopted. We present averaged mAUC and mDP in 4 datasets, DTB70 (Li & Yeung, 2017), UAVDT (Du et al., 2018), UAV20L (Mueller et al., 2016), and UAV123 (Mueller et al., 2016). Compared with KF, our learning framework holds the obvious advantage in complex UAV tracking scenes. For more exhaustive comparison, please refer to Appendix E nad Appendix F.

6.4 Real-world Tests

We further deploy SiamMask (Wang et al., 2019) ($\sim$11FPS) and SiamRPN++$_{\mathrm{Mob}}$ (Li et al., 2019) ($\sim$15FPS) with PVT++ on a UAV with Nvidia Jetson AGX Xavier as onboard processor. The result is shown in Fig. 5. 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. We present more real-world tests in Appendix L.

7 Discussions

Limitation. (1) Consider SiamRPN++_Mob with $\sim$ 45ms latency, PVT++ introduces an extra latency of $\sim$ 5ms per frame, which is higher than KF (2 $\sim$ 3ms), slightly affecting the performance (further discussion in Appendix J). (2) For e-LAE, the performance is usually influenced by the state of the hardware, which requires multiple runs for a proper assessment.

Conclusion. In this work, we present a simple end-to-end learnable framework for latency-aware visual tracking, PVT++, which practically eliminates onboard latency. PVT++ integrates a predictor module that predicts objects' future location based on both motion and visual appearance. Jointly optimizing the predictor and the tracker yields a strong performance. Extensive evaluations on robotics platform from the challenging aerial perspective show the effectiveness of PVT++ framework, which improves the offline tracker by up to **60**% in the online setting. We hope that our approach can facilitate more research on predictive visual tracking for real-world tracking tasks.

REFERENCES

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)*, pp. 386–398, 2019.

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)*, pp. 850–865, 2016.

Goutam Bhat, Martin Danelljan, L. Gool, and R. Timofte. Learning Discriminative Model Prediction for Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6181–6190, 2019.

D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual Object Tracking Using Adaptive Correlation Filters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2544–2550, 2010.

Rogerio 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)*, pp. 229–236, 2019.

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

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)*, pp. 15457–15466, 2021b.

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

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, 2020a.

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)*, pp. 6668–6677, 2020b.

M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Learning Spatially Regularized Correlation Filters for Visual Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4310–4318, 2015.

M. Danelljan, L. Van Gool, and R. Timofte. Probabilistic Regression for Visual Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7181–7190, 2020.

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)*, pp. 6638–6646, 2017.

Martin Danelljan, Goutam Bhat, F. Khan, and M. Felsberg. ATOM: Accurate Tracking by Overlap Maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4655–4664, 2019.

Dawei 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)*, pp. 370–386, 2018.

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)*, pp. 300–317, 2020.

Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. 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)*, pp. 5369–5378, 2019.

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)*, pp. 510–516, 2021.

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.

Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6269–6277, 2020.

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)*, pp. 9543–9552, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2018.

J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

L. Huang, Xin Zhao, and K. Huang. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019a.

Ziyuan Huang, C. Fu, Y. Li, Fuling Lin, and Peng Lu. Learning Aberrance Repressed Correlation Filters for Real-Time UAV Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2891–2900, 2019b.

Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME-Joural of Basic Engineering*, pp. 35–45, 1960.

B. Li, J. Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High Performance Visual Tracking with Siamese Region Proposal Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8971–8980, 2018.

B. Li, Wei Wu, Q. Wang, Fangyi Zhang, Junliang Xing, and J. Yan. SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4277–4286, 2019.

Bowen Li, Changhong Fu, Fangqiang Ding, Junjie Ye, and Fuling Lin. ADTrack: Target-Aware Dual Filter Learning for Real-Time Anti-Dark UAV Tracking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 496–502, 2021a.

Bowen Li, Yiming Li, Junjie Ye, Changhong Fu, and Hang Zhao. Predictive Visual Tracking: A New Benchmark and Baseline Approach. *arXiv preprint arXiv:2103.04508*, pp. 1–8, 2021b.

F. Li, C. Fu, Fuling Lin, Y. Li, and Peng Lu. Training-Set Distillation for Real-Time UAV Object Tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9715–9721, 2020a.

M. Li, Yu-Xiong Wang, and D. Ramanan. Towards Streaming Perception. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 473–488, 2020b.

Siyi Li and Dit-Yan Yeung. Visual Object Tracking for Unmanned Aerial Vehicles: A Benchmark and New Motion Models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4140–4146, 2017.

Y. Li, C. Fu, Fangqiang Ding, Ziyuan Huang, and Geng Lu. AutoTrack: Towards High-Performance Visual Tracking for UAV with Automatic Spatio-Temporal Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11920–11929, 2020c.

Y. Li, C. Fu, Ziyuan Huang, Yinqiang Zhang, and Jia Pan. Keyfilter-Aware Real-Time UAV Object Tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 193–199, 2020d.

Q. Liu, Zhenyu He, Xin Li, and Yuan Zheng. PTB-TIR: A Thermal Infrared Pedestrian Tracking Benchmark. *IEEE Transactions on Multimedia*, 22:666–675, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–18, 2018.

Matthias Mueller, Neil Smith, and Bernard Ghanem. A Benchmark and Simulator for UAV Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 445–461, 2016.

M. Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 300–317, 2018.

Yuya Nishida, Takashi Sonoda, Shinsuke Yasukawa, Kazunori Nagano, Mamoru Minami, Kazuo Ishii, and Tamaki Ura. Underwater Platform for Intelligent Robotics and its Application in Two Visual Tracking Systems. *Journal of Robotics and Mechatronics*, 30(2):238–247, 2018.

Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer Meets Tracker: Exploiting Temporal Context for Robust Visual Tracking. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1571–1580, 2021.

Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast Online Object Tracking and Segmentation: A Unifying Approach. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1328–1338, 2019.

Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 12549–12556, 2020.

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)*, pp. 1–8, 2022.

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.

Zheng Zhu, Q. Wang, B. Li, Wei Wu, J. Yan, and W. Hu. Distractor-aware Siamese Networks for Visual Object Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

APPENDIX

A OVERVIEW

To make our end-to-end predictive visual tracking framework (PVT++) reproducible, we present the detailed configuration in Appendix B, covering the specific model structure, the training settings (with specific hyper-parameters), and the inference settings. Moreover, we provide the PVT++ code library and official models to ensure reproducability. For clear reference of the notations used in method section, we provide a notation table in Appendix C. In Appendix D, we display representative qualitative visualization results from the authoritative datasets, UAV123 (Mueller et al., 2016), UAV20L (Mueller et al., 2016), DTB70 (Li & Yeung, 2017), and UAVDT (Du et al., 2018), where the superiority of our PVT++ is clearly shown. In Appendix E and Appendix F, we present detailed results comparison between KF (Kalman, 1960) and PVT++ to better demonstrate the superiority of our method. We also tried to fuse the motion and visual cues earlier in Appendix G, where we give an analysis to the strategy adopted in PVT++. The full attribute-based results from all the four datasets (Mueller et al., 2016; Li & Yeung, 2017; Du et al., 2018) are reported in Appendix H, where we exhaustively analyse the specific advantages of two modalities for prediction under various UAV tracking challenges. The training process of different PVT++ models is visualized in Appendix I, where we present the loss curves to indicate the converging process. The extra latency introduced by the PVT++ predictor modules is unavoidable, which can have some negative effect to online performance. We provide such analysis in Appendix J. We further find PVT++ is capable of converging well in smaller training set (using only 3563 videos from Imagenet VID (Russakovsky et al., 2015)), which is shown in Appendix K. Finally, we present additional real-world tests in Appendix L, covering more target objects and tracking scenes.

B DETAILED CONFIGURATION

Specific Model Structure. Corresponding to Fig. 3 in the paper, we present the detailed model structure of each layer in Table I. Consider B batch inputs and k history frames, the output sizes are also shown in Table I for clear reference. Subscripts are used to distinguish between different layers, *i.e.*, $\cdot_{\mathrm{t}}$ denotes encoding layer for template feature, $\cdot_{\mathrm{s}}$ denotes encoding layer for search feature, $\cdot_{\mathrm{e}}$ denotes encoding layer for the similarity map. $\cdot_{\mathrm{a}}$ represents the auxiliary branch.

Training Settings. All the predictive modules need temporal video data for training. However, to our disappointment, existing training pipeline (Li et al., 2019) takes a detection-like paragdim. Basically, the raw search patches are *independently* cropped from the object center location, then the random shift, padding are applied to generated the training search patch. In this case, the training patches from consecutive frames actually contain no temporal information.

To solve this, we construct a new pipeline termed as dynamic temporal training. The search patch from f_j-th frame is cropped around the object's center location in the previous frame $\mathcal{I}_{f_{j-1}}$, so that past motion $\mathbf{M}_{\phi(f)}$ and past search patch $\mathbf{X}_{\phi(f)}$ correspond to each other and contain real temporal information from $\mathcal{I}_{f_{j-k+1}}$ to $\mathcal{I}_{f_j}$.

Remark 1: The new training pipeline is dynamic, *i.e.*, $[f_{j-k}, f_{j-k+1}, \cdots, f_j]$ can be adjusted as hyper-parameters to fit different models' different latency.

All the PVT++ models are optimized by AdamW (Loshchilov & Hutter, 2018). The motion predictor is trained for 100 epochs with a base learning rate equalling to 0.03, which is multiplied by 0.1 at epoch 30 and 80. The visual and multi-modal predictors are trained for 300 epochs with a base learning rate of 0.003, which is multiplied by 0.1 at epoch 200. In all the three base trackers, $\mathcal{P}_{\mathrm{V}}$ and $\mathcal{P}_{\mathrm{MV}}$ both take the visual feature from the neck to implement vision-aided prediction. During joint training, the tracker backbone is fixed and the tracker neck, together with the head are freed in the first 20 epochs with a small learning rate of 10^{-4}.

A "fast" tracker may only need to predict future three frames to compensate for its latency, while a "slow" one may have to output ten future state. To make this possible, the second last layer of PVT++ predictive decoder is N parallel fully connected layers for predicting N future state, *i.e.*, future $1 \sim N$ frames. Therefore, different models vary in the pre-defined N and Δ_f during training. we set $N = 3, \Delta_f = [1 : 3]$ for SiamRPN++_Mob (Li et al., 2019), $N = 12, \Delta_f = [1 : 12]$ for

Table I: Detailed structure and output sizes of PVT++ models. We use subscript to distinguish between different layers. The output sizes correspond to B batch input.

Branch	Layer	Kernel	In. Channel	Out. Channel	Out. Size
Motion	FC	-	8	32	$B \times k \times 32$
	1D Conv	3	32	32	$B \times k \times 32$
	Avg. Pool	-	32	32	$B \times 32$
Visual	2D Conv_t	3×3	256	64	$B \times k \times 64 \times 29 \times 29$
	2D Conv_s	3×3	256	64	$B \times k \times 64 \times 25 \times 25$
	2D Conv_e	1×1	64	64	$B \times k \times 64 \times 25 \times 25$
	3D Conv	$3 \times 3 \times 3$	64	64	$B \times k \times 64 \times 25 \times 25$
	Avg. Pool	-	64	64	$B \times 64$
	2D Conv_a	1×1	64	64	$B \times k \times 64 \times 25 \times 25$
	2D Conv_a	1×1	64	4	$B \times k \times 4 \times 25 \times 25$
	Avg. Pool_a	-	4	4	$B \times k \times 4$
Shared	FC	-	[32, 64, 96]	32	$B \times 32$
	FC	-	32	32	$B \times N \times 32$
	FC	-	32	4	$B \times N \times 4$

Table II: Attribute-based analysis of the three trackers with PVT++ models in DTB70 (Li & Yeung, 2017) dataset.

Tracker		$\text{SiamRPN++}_\text{Mob}$ (21FPS)				$\text{SiamRPN++}_\text{Res}$ (5FPS)				SiamMask (12FPS)			
Metric	Att.	N/A	$\mathcal{P}_\text{M}$	$\mathcal{P}_\text{V}$	$\mathcal{P}_\text{MV}$	N/A	$\mathcal{P}_\text{M}$	$\mathcal{P}_\text{V}$	$\mathcal{P}_\text{MV}$	N/A	$\mathcal{P}_\text{M}$	$\mathcal{P}_\text{V}$	$\mathcal{P}_\text{MV}$
AUC@La0	ARV	0.330	0.386	0.349	0.418	0.156	0.233	0.214	0.253	0.247	0.375	0.291	0.393
	BC	0.257	0.330	0.276	0.319	0.079	0.077	0.102	0.102	0.168	0.264	0.202	0.167
	DEF	0.357	0.410	0.358	0.438	0.144	0.217	0.198	0.241	0.253	0.398	0.287	0.364
	FCM	0.277	0.373	0.333	0.376	0.091	0.144	0.122	0.138	0.195	0.327	0.258	0.301
	IPR	0.302	0.368	0.324	0.387	0.133	0.187	0.169	0.204	0.217	0.346	0.256	0.316
	MB	0.198	0.305	0.277	0.321	0.056	0.073	0.069	0.085	0.147	0.236	0.187	0.254
	OCC	0.280	0.337	0.281	0.304	0.149	0.214	0.204	0.224	0.233	0.290	0.285	0.274
	OPR	0.278	0.314	0.334	0.439	0.161	0.158	0.208	0.225	0.202	0.360	0.265	0.362
	OV	0.292	0.405	0.372	0.399	0.054	0.099	0.076	0.102	0.168	0.227	0.258	0.289
	SV	0.354	0.470	0.419	0.489	0.145	0.187	0.192	0.220	0.278	0.435	0.347	0.418
	SOA	0.238	0.301	0.261	0.302	0.140	0.196	0.184	0.200	0.227	0.326	0.275	0.315
DP@La0	ARV	0.340	0.466	0.385	0.498	0.101	0.220	0.171	0.234	0.247	0.474	0.333	0.472
	BC	0.352	0.477	0.396	0.498	0.118	0.106	0.141	0.139	0.228	0.385	0.291	0.237
	DEF	0.374	0.512	0.398	0.525	0.083	0.203	0.144	0.214	0.246	0.509	0.326	0.449
	FCM	0.363	0.517	0.470	0.525	0.106	0.188	0.156	0.171	0.241	0.456	0.353	0.414
	IPR	0.349	0.475	0.398	0.495	0.124	0.212	0.170	0.224	0.236	0.454	0.310	0.400
	MB	0.246	0.418	0.379	0.453	0.051	0.110	0.090	0.088	0.167	0.349	0.248	0.327
	OCC	0.408	0.496	0.426	0.459	0.223	0.327	0.316	0.344	0.361	0.439	0.458	0.404
	OPR	0.213	0.312	0.317	0.453	0.083	0.083	0.113	0.127	0.128	0.382	0.224	0.357
	OV	0.413	0.590	0.564	0.586	0.062	0.166	0.101	0.161	0.222	0.363	0.385	0.439
	SV	0.366	0.569	0.467	0.569	0.123	0.186	0.180	0.208	0.287	0.528	0.402	0.492
	SOA	0.333	0.432	0.379	0.447	0.217	0.306	0.295	0.302	0.340	0.479	0.429	0.462

SiamRPN++_Res (Li et al., 2019), and $N = 6, \Delta_f = [1:6]$ for SiamMask (Wang et al., 2019). Note that these hyper-parameter are roughly determined by the averaged latency of the base trackers.

Inference Settings. During inference, when $f_{j+1}-$th frame comes, the predictor $\mathcal{P}$ first conducts $(f_{j+1} - f_j)$ to $f_{j+1} + N$ frames prediction with $k = 3$ past frames information, then the tracker processes $f_{j+1} - th$ frame and updates the history information (motion and visual).

Note that we take the latency of both tracker and predictor modules into account in the online evaluation.

C Complete Notation Reference Table

We provide the important notations, their meaning, and dimension in Table III, for clear reference.

Table III: List of the important notations in this work.

Symbol	Meaning	Dimension
f	World frame number	$\mathbb{R}$
$\mathcal{I}_f$	f-th image frame	$\mathbb{R}^{W\times H\times 3}$
j	Serial number of the processed frame	$\mathbb{R}$
f_j	World frame id of the processed j-th frame	$\mathbb{R}$
t_f^{W}	World timestamp	$\mathbb{R}$
$t_{f_j}^{\mathrm{T}}$	Tracker timestamp	$\mathbb{R}$
$\phi(f), \phi(f)_e$	Input frame id to be paired with frame f	$\mathbb{R}$
σ	Permitted latency during evaluation	$\mathbb{R}$
$\mathbf{r}_f = [x_f, y_f, w_f, h_f]$	Raw output by the tracker in frame f	$\mathbb{R}^{1\times 4}$
$\hat{\mathbf{b}}_f = [\hat{x}_f, \hat{y}_f, \hat{w}_f, \hat{h}_f]$	Final output bounding box to be evaluated	$\mathbb{R}^{1\times 4}$
$\mathcal{T}$	Tracker model	–
$\mathcal{P}$	Predictor model	–
$\mathbf{m}_{f_j}$	Normalized input motion from frame f_{j-1} to f_j	$\mathbb{R}^{1\times 4}$
$\mathbf{p}_{f_j}$	Average moving speed from frame f_{j-k+1} to f_j	$\mathbb{R}^{1\times 4}$
$\hat{\mathbf{m}}_f$	Predicted motion from frame $\phi(f)$ to f	$\mathbb{R}^{1\times 4}$
$\mathbf{m}_f$	Ground-truth motion from frame $\phi(f)$ to f	$\mathbb{R}^{1\times 4}$
Δ_f	Frame interval between the latest frame and the f-th frame	$\mathbb{R}$
$\Delta_{\hat{x}}(f), \Delta_{\hat{y}}(f)$	Predicted box distance between the f-th and $\phi(f)$-th frame	$\mathbb{R}$
$\Delta_x(f_j), \Delta_y(f_j)$	Distance from $\mathbf{r}_{f_j}$ to $\mathbf{r}_{f_{j-1}}$	$\mathbb{R}$
$\mathbf{x}_{\phi(f)}$	Search patch feature in frame $\phi(f)$ from tracker backbone	$\mathbb{R}^{C\times W\times H}$
$\mathbf{z}$	Template feature from tracker backbone	$\mathbb{R}^{C\times a\times a}$
$k(=3)$	Number of past frames for the history information	$\mathbb{R}$
N	Number of the parallel FC layers in the predictive decoder	$\mathbb{R}$

Table IV: Attribute-based analysis of the three trackers with PVT++ models in UAVDT (Du et al., 2018) dataset.

Tracker		SiamRPN++$_{\mathrm{Mob}}$ (21FPS)				SiamRPN++$_{\mathrm{Res}}$ (5FPS)				SiamMask (12FPS)			
Metric	Att.	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$
AUC@La0	BC	0.448	0.461	0.504	0.505	0.332	0.410	0.375	0.445	0.404	0.465	0.488	0.520
	CR	0.450	0.495	0.520	0.535	0.296	0.371	0.402	0.452	0.425	0.503	0.498	0.522
	OR	0.438	0.481	0.538	0.549	0.318	0.389	0.416	0.477	0.404	0.491	0.504	0.541
	SO	0.494	0.549	0.525	0.545	0.318	0.420	0.361	0.457	0.468	0.536	0.495	0.540
	IV	0.539	0.578	0.588	0.599	0.382	0.495	0.459	0.537	0.475	0.558	0.563	0.596
	OB	0.525	0.542	0.568	0.589	0.382	0.460	0.408	0.498	0.471	0.542	0.527	0.560
	SV	0.490	0.505	0.584	0.586	0.366	0.422	0.406	0.484	0.438	0.526	0.541	0.566
	LO	0.422	0.521	0.436	0.511	0.320	0.379	0.368	0.429	0.389	0.421	0.494	0.520
DP@La0	BC	0.659	0.666	0.733	0.727	0.591	0.637	0.647	0.671	0.628	0.672	0.718	0.731
	CR	0.643	0.684	0.720	0.732	0.462	0.585	0.572	0.645	0.620	0.702	0.696	0.712
	OR	0.638	0.681	0.753	0.764	0.515	0.619	0.606	0.688	0.612	0.709	0.723	0.752
	SO	0.779	0.815	0.793	0.814	0.645	0.711	0.706	0.759	0.803	0.818	0.787	0.819
	IV	0.777	0.811	0.835	0.848	0.657	0.747	0.755	0.801	0.743	0.797	0.817	0.829
	OB	0.772	0.778	0.822	0.846	0.676	0.714	0.700	0.766	0.756	0.802	0.801	0.813
	SV	0.680	0.691	0.796	0.794	0.581	0.618	0.622	0.684	0.650	0.729	0.763	0.783
	LO	0.569	0.717	0.585	0.694	0.504	0.554	0.566	0.608	0.571	0.590	0.696	0.711

D VISUALIZATION

We present some typical tracking visualization in Fig. I. The sequences, *ManRunning2*, *Paragliding5*, *Wakeboarding1*, and *Wakeboarding2* are from DTB70 (Li & Yeung, 2017).*S0303*, *S0304*, *S0310*, and *S1604* are from UAVDT (Du et al., 2018). In UAV20L and UAV123 (Mueller et al., 2016), we also present *car3*, *car17*, *group2_2*, and *uav1_2*. With extremely limited onboard computation, the original trackers (red dashed boxes) will easily fail due to high latency. Once coupled

Figure I: Representative sequences from authoritative UAV tracking datasets, DTB70 (Li & Yeung, 2017), UAVDT (Du et al., 2018), UAV20L (Mueller et al., 2016), and UAV123 (Mueller et al., 2016). We use dashed red lines to demonstrate the original trackers, which are severely affected by onboard latency. Coupled with our PVT++ ($\mathcal{P}_{\mathrm{MV}}$), the robustness can be significantly improved (solid red boxes). Green boxes denote ground-truth. Some typical sequences are also made into supplementary video for better reference.

Table V: Per dataset results of different predictor modules. For all the three base trackers in various datasets, our PVT++ outperforms traditional KF (Kalman, 1960).

Tracker	Dataset / Pred.	DTB70 AUC@La0	DTB70 DP@La0	UAVDT AUC@La0	UAVDT DP@La0	UAV20L AUC@La0	UAV20L DP@La0	UAV123 AUC@La0	UAV123 DP@La0
SiamRPN++$_{\mathrm{Mob}}$ (21FPS)	N/A	0.305	0.387	0.494	0.719	0.448	0.619	0.472	0.678
	KF	0.349	0.482	0.527	0.737	0.458	0.624	0.515	0.712
	PVT$_{\mathrm{MV}}$	**0.399**	**0.536**	**0.576**	**0.807**	**0.508**	**0.697**	**0.537**	**0.741**
SiamRPN++$_{\mathrm{Res}}$ (5FPS)	N/A	0.247	0.313	0.455	0.703	0.405	0.571	0.436	0.639
	KF	0.294	0.407	0.535	0.758	0.436	0.582	0.499	0.679
	PVT$_{\mathrm{MV}}$	**0.342**	**0.463**	**0.566**	**0.797**	**0.469**	**0.644**	**0.536**	**0.749**
SiamMask (12FPS)	N/A	0.136	0.159	0.351	0.594	0.31	0.434	0.349	0.505
	KF	0.189	0.232	0.451	0.667	0.387	0.528	0.415	0.582
	PVT$_{\mathrm{MV}}$	**0.205**	**0.256**	**0.488**	**0.726**	**0.416**	**0.568**	**0.442**	**0.619**

with our PVT++ ($\mathcal{P}_{\mathrm{MV}}$), the models (solid red boxes) are much more robust. We use greed boxes to denote ground-truth for clear reference.

E Prediction Quantitative Comparison

To provide a thorough quantitative comparison of the predictor performance, we reported the results per dataset in Table V. We observe that for different tracker models in various benchmarks,

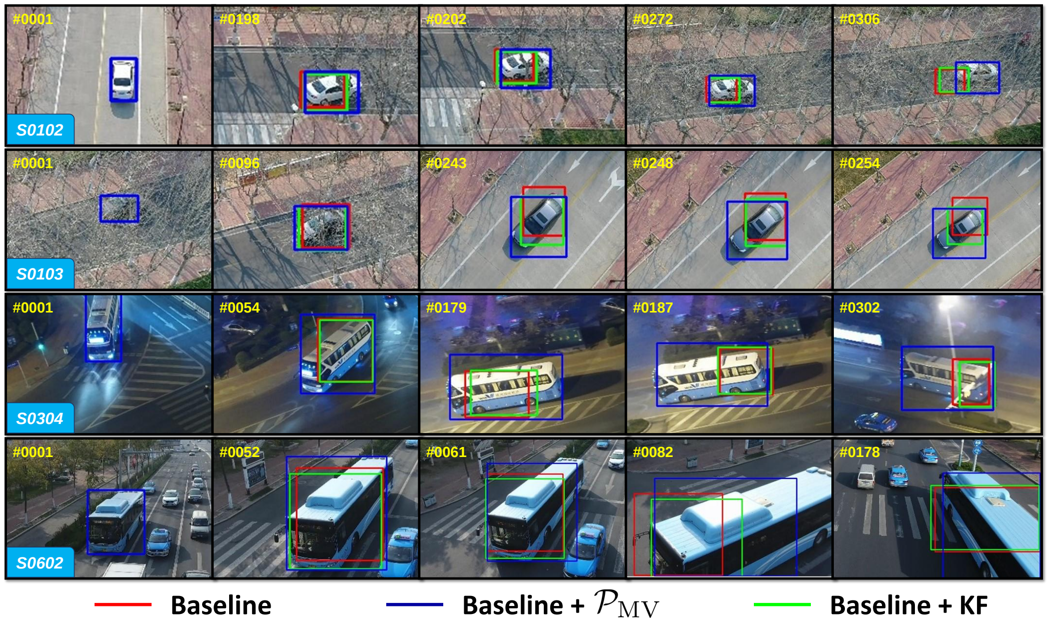

Figure II: Prediction comparison from UAVDT (Du et al., 2018). We use red lines to demonstrate the original trackers, green for the KF (Kalman, 1960) prediction, and blue for PVT++ prediction. Compared to KF, PVT++ is better at handling challenges like rotation, scale variation, and view point change.

Table VI: Results comparison between two fusion strategy. $\mathcal{P}_{\mathrm{MV}}$ denotes our default PVT++, the modalities fuse after independent temporal interaction (late fusion). $\mathcal{P}^{\dagger}_{\mathrm{MV}}$ indicates that the two cues fuse before temporal interaction (early fusion).

Pred.	DTB70 AUC@La0	DTB70 DP@La0	UAVDT AUC@La0	UAVDT DP@La0
N/A	0.305	0.387	0.494	0.719
$\mathcal{P}_{\mathrm{MV}}$ (late fuse)	0.399	0.536	0.576	0.807
$\mathcal{P}^{\dagger}_{\mathrm{MV}}$ (early fuse)	0.370	0.498	0.571	0.800

our PVT++ ($\mathcal{P}_{\mathrm{MV}}$) is more robust than prior solutions (Li et al., 2021b; 2020b), which adopted traditional KF (Kalman, 1960).

F Prediction Qualitative Comparison

We also present some qualitative comparison between KF (Kalman, 1960) and PVT++ in Fig. II. To provide a valid comparison of the prediction results, we set the latency of the models the same, which all adopt SiamRPN++_Mob (∼21FPS). We find that compared with KF, PVT++ is better at predicting in-plane rotation (*S0103*) and view point change (*S0304*, *S0602*).

G Fusion Strategy Comparison

As introduced in the paper, inside PVT++, the three modules, *Feature encoder*, *temporal interaction*, and *predictive decoder* run one after another. For the default setting, the fusion of the motion and visual cues happens after *temporal interaction*, using the concatenate function. Here, we also tried to integrate the two modality earlier before *temporal interaction* and right after *feature encoder*, still adopting concatenation. The results comparison of two strategies is shown in Table VI, where we find both are effective and the late fusion is better.

Table VII: Attribute-based analysis of the three trackers with PVT++ models in UAV20L (Mueller et al., 2016) dataset.

Tracker		SiamRPN++$_{\mathrm{Mob}}$ (21FPS)				SiamRPN++$_{\mathrm{Res}}$ (5FPS)				SiamMask (12FPS)			
Metric	Att.	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$
AUC@La0	SV	0.437	0.470	0.483	0.500	0.300	0.395	0.392	0.410	0.395	0.437	0.420	0.461
	ARC	0.425	0.411	0.438	0.451	0.291	0.352	0.360	0.371	0.373	0.409	0.392	0.438
	LR	0.267	0.354	0.344	0.352	0.215	0.295	0.276	0.279	0.244	0.263	0.275	0.290
	FM	0.410	0.357	0.394	0.418	0.269	0.304	0.325	0.315	0.319	0.375	0.329	0.442
	FOC	0.256	0.272	0.234	0.241	0.170	0.227	0.184	0.164	0.221	0.237	0.231	0.255
	POC	0.418	0.480	0.463	0.478	0.286	0.379	0.380	0.396	0.378	0.417	0.430	0.441
	OV	0.438	0.512	0.476	0.492	0.272	0.356	0.394	0.405	0.377	0.428	0.448	0.462
	BC	0.225	0.258	0.229	0.250	0.119	0.215	0.153	0.159	0.189	0.198	0.210	0.210
	IV	0.452	0.414	0.470	0.491	0.303	0.393	0.379	0.403	0.426	0.437	0.382	0.443
	VC	0.472	0.450	0.466	0.488	0.302	0.339	0.377	0.384	0.395	0.436	0.420	0.475
	CM	0.431	0.463	0.475	0.491	0.297	0.393	0.388	0.406	0.391	0.432	0.412	0.452
	SO	0.482	0.519	0.557	0.567	0.399	0.531	0.477	0.491	0.487	0.519	0.438	0.492
DP@La0	SV	0.600	0.630	0.662	0.683	0.417	0.544	0.536	0.556	0.552	0.588	0.581	0.627
	ARC	0.591	0.562	0.606	0.624	0.408	0.487	0.486	0.503	0.524	0.558	0.550	0.603
	LR	0.444	0.545	0.539	0.548	0.388	0.483	0.465	0.456	0.422	0.414	0.458	0.465
	FM	0.631	0.548	0.595	0.625	0.417	0.464	0.495	0.476	0.518	0.573	0.524	0.667
	FOC	0.469	0.473	0.436	0.428	0.358	0.423	0.358	0.324	0.425	0.420	0.431	0.459
	POC	0.585	0.654	0.648	0.669	0.410	0.530	0.531	0.548	0.540	0.570	0.606	0.613
	OV	0.597	0.683	0.658	0.679	0.356	0.473	0.518	0.540	0.529	0.578	0.618	0.630
	BC	0.426	0.440	0.399	0.434	0.284	0.398	0.304	0.295	0.378	0.349	0.390	0.385
	IV	0.628	0.560	0.649	0.686	0.428	0.551	0.503	0.539	0.595	0.590	0.545	0.617
	VC	0.616	0.571	0.605	0.631	0.364	0.420	0.452	0.477	0.518	0.551	0.546	0.611
	CM	0.599	0.629	0.660	0.681	0.417	0.544	0.534	0.553	0.550	0.588	0.580	0.626
	SO	0.604	0.652	0.719	0.734	0.498	0.645	0.594	0.609	0.610	0.648	0.559	0.619

Table VIII: Attribute-based analysis of the three trackers with PVT++ models in UAV123 (Mueller et al., 2016) dataset.

Tracker		SiamRPN++$_{\mathrm{Mob}}$ (21FPS)				SiamRPN++$_{\mathrm{Res}}$ (5FPS)				SiamMask (12FPS)			
Metric	Att.	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$	N/A	$\mathcal{P}_{\mathrm{M}}$	$\mathcal{P}_{\mathrm{V}}$	$\mathcal{P}_{\mathrm{MV}}$
AUC@La0	SV	0.456	0.518	0.488	0.514	0.338	0.423	0.383	0.427	0.420	0.509	0.480	0.518
	ARC	0.413	0.496	0.468	0.491	0.315	0.402	0.365	0.406	0.398	0.498	0.467	0.510
	LR	0.291	0.357	0.328	0.350	0.179	0.264	0.214	0.256	0.257	0.364	0.324	0.257
	FM	0.373	0.430	0.461	0.482	0.261	0.316	0.307	0.341	0.333	0.425	0.422	0.447
	FOC	0.254	0.317	0.270	0.306	0.191	0.251	0.214	0.246	0.242	0.325	0.284	0.303
	POC	0.401	0.436	0.402	0.446	0.284	0.373	0.335	0.374	0.363	0.449	0.426	0.459
	OV	0.442	0.489	0.488	0.516	0.289	0.394	0.368	0.407	0.403	0.504	0.476	0.492
	BC	0.254	0.293	0.247	0.296	0.188	0.258	0.215	0.247	0.248	0.360	0.307	0.309
	IV	0.365	0.421	0.423	0.465	0.310	0.379	0.352	0.381	0.378	0.480	0.441	0.466
	VC	0.459	0.552	0.506	0.558	0.322	0.409	0.387	0.432	0.407	0.534	0.499	0.548
	CM	0.466	0.542	0.514	0.535	0.319	0.421	0.381	0.422	0.420	0.529	0.502	0.522
	SO	0.478	0.497	0.444	0.459	0.362	0.462	0.382	0.435	0.434	0.492	0.464	0.514
DP@La0	SV	0.657	0.714	0.679	0.710	0.488	0.599	0.594	0.537	0.614	0.711	0.671	0.720
	ARC	0.602	0.689	0.651	0.678	0.453	0.575	0.502	0.561	0.588	0.701	0.656	0.715
	LR	0.548	0.595	0.568	0.586	0.392	0.488	0.471	0.438	0.510	0.621	0.554	0.637
	FM	0.517	0.591	0.617	0.646	0.323	0.417	0.368	0.429	0.450	0.588	0.564	0.609
	FOC	0.497	0.550	0.489	0.533	0.387	0.460	0.406	0.448	0.460	0.569	0.505	0.541
	POC	0.614	0.630	0.586	0.640	0.440	0.556	0.497	0.542	0.553	0.653	0.619	0.664
	OV	0.632	0.670	0.674	0.715	0.372	0.533	0.467	0.533	0.556	0.701	0.653	0.685
	BC	0.474	0.475	0.436	0.489	0.407	0.470	0.411	0.444	0.473	0.587	0.512	0.526
	IV	0.546	0.594	0.586	0.644	0.447	0.541	0.521	0.486	0.550	0.674	0.623	0.664
	VC	0.654	0.743	0.681	0.746	0.443	0.575	0.512	0.586	0.587	0.735	0.683	0.744
	CM	0.668	0.748	0.713	0.735	0.440	0.587	0.514	0.573	0.606	0.737	0.699	0.734
	SO	0.714	0.703	0.625	0.647	0.554	0.681	0.568	0.639	0.650	0.691	0.671	0.724

H Full Attribute-based Analysis

We present full attribute-based analysis in Table II, Table IV, Table VII, and Table VIII. Following the original work (Li & Yeung, 2017), we report results on aspect ratio variation (ARV), background

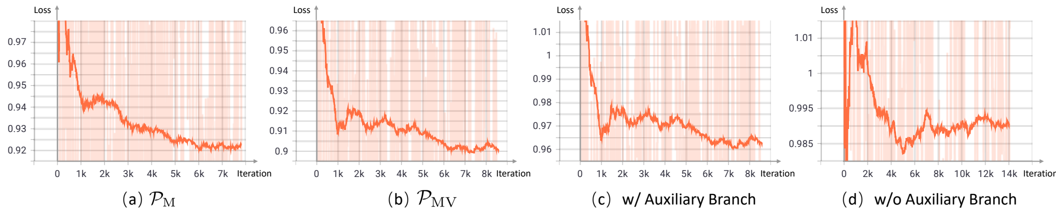

Figure III: Training loss curves of PVT++ models. Coupled with visual feature, $\mathcal{P}_{\mathrm{MV}}$ can better learn to predict than $\mathcal{P}_{\mathrm{M}}$, thus the loss is observed to be smaller. Without auxiliary branch, the loss curve is less smooth, indicating the importance of $\mathcal{A}$.

Table IX: Effect of extra latency brought by PVT++ in UAVDT (Du et al., 2018) dataset. We use $\cdot^{\dagger}$ to indicate neglecting the latency. With ∼5ms/frame extra time, the performance is slightly lower.

Model	Tracker			Tracker+$\mathcal{P}_{\mathrm{MV}}^{\dagger}$			Tracker+$\mathcal{P}_{\mathrm{MV}}$		
Metric	mAUC$_{\Delta\%}$	mDP$_{\Delta\%}$	Latency	mAUC$_{\Delta\%}$	mDP$_{\Delta\%}$	Latency	mAUC$_{\Delta\%}$	mDP$_{\Delta\%}$	Latency
Result	0.494$_{+0.00}$	0.719$_{+0.00}$	44.5ms	0.587$_{+18.8}$	0.825$_{+14.7}$	44.5ms	0.576$_{+16.6}$	0.807$_{+12.2}$	50.0ms

Table X: Performance of PVT++ models trained with different datasets. Full denotes ∼9,000 videos from VID (Russakovsky et al., 2015), LaSOT (Fan et al., 2019), and GOT-10k (Huang et al., 2019a). VID indicates using only ∼3,000 videos from VID (Russakovsky et al., 2015). AVG means average results on the four test datasets. Since PVT++ utilizes the trained tracking models, We observe the training are not very sensitive to the scale of training set.

Dataset		DTB70		UAVDT		UAV20L		UAV123		AVG	
PVT++	Training	mAUC	mDP	mAUC	mDP	mAUC	mDP	mAUC	mDP	mAUC	mDP
$\mathcal{P}_{\mathrm{V}}$	Full	0.352	0.472	0.564	0.799	0.488	0.675	0.504	0.703	**0.477**	0.662
	VID	0.362	0.483	0.519	0.752	0.497	0.694	0.513	0.731	0.473	**0.665**
$\mathcal{P}_{\mathrm{MV}}$	Full	0.399	0.536	0.576	0.807	0.508	0.697	0.537	0.741	**0.505**	**0.695**
	VID	0.405	0.554	0.53	0.757	0.511	0.701	0.534	0.745	0.495	0.689

clutter (BC), deformation (DEF), fast camera motion (FCM), in-plane rotation (IPR), motion blur (MB), occlusion (OCC), out-of-plane rotaTion (OPR), out-of-view (OV), scale variation (SV), and similar object around (SOA) in Table II. As shown in Table IV, results on background clutter (BC), camera rotation (CR), object rotation (OR), small object (SO), illumination variation (IV), object blur (OB), scale variation (SV), and large occlusion (LO), are reported for UAVDT (Du et al., 2018). For UAV20L and UAV123 (Mueller et al., 2016), we present results on scale variation (SV), aspect ratio change (ARC), low resolution (LR), fast motion (FM), full occlusion (FOC), partial occlusion (POC), out-of-view (OV), background clutter (BC), illumination variation (IV), viewpoint change (VC), camera motion (CM), and similar object (SO) in Table VII and Table VIII, respectively.

We observe that the two modalities has their own advantage in different UAV tracking challenges. In general, motion predictor is better than visual predictor, and the joint model $\mathcal{P}_{\mathrm{MV}}$ is the most robust.

I Training Visualization

The training loss curves of PVT++ models with SiamRPN++$_{\mathrm{Mob}}$ (Li et al., 2019) is shown in Fig. III. Compared with motion predictor $\mathcal{P}_{\mathrm{M}}$, the joint predictor $\mathcal{P}_{\mathrm{MV}}$ can better learn to predict, resulting in smaller training loss. We also compared the losses from models with (c) or without (d) the auxiliary branch $\mathcal{A}$. Without $\mathcal{A}$, the loss curve is less smooth, indicating that the model can't converge well.

J Effect of Extra Latency

PVT++ will bring a bit extra latency during online perception, which is negative for the performance. As shown in Table IX, the latency of original tracker (Li et al., 2019) is about 45 ms/frame. Ignoring

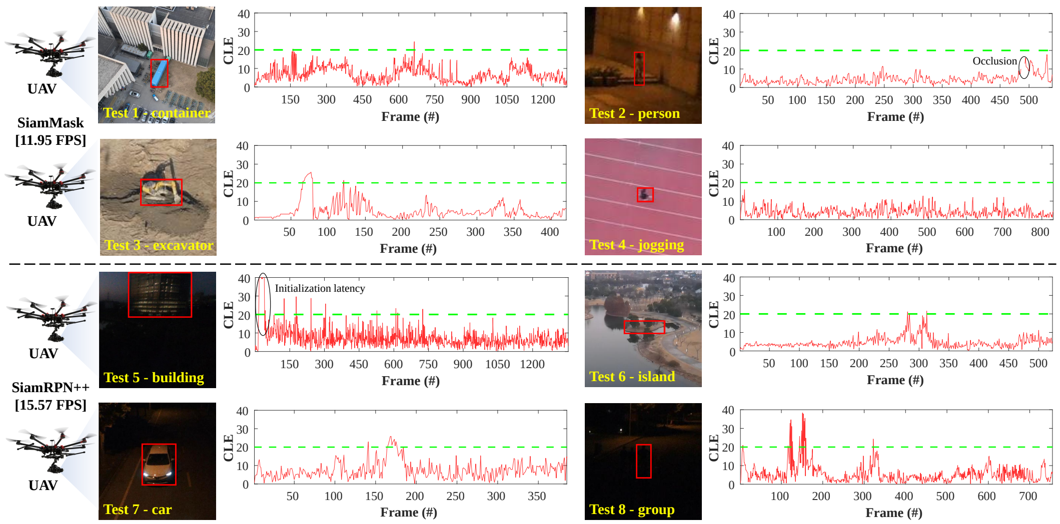

Figure IV: Eight real-world tests of PVT++ on non-real-time trackers, SiamMask (Wang et al., 2019) and SiamRPN++_Mob (Li et al., 2019). We present the tracking scenes, the target objects, and center location error (CLE) in the figure. Under various challenges like aspect ration change, illumination variation, low resolution, PVT++ maintains its robustness, with CLE below 20 pixels in most frames.

the predictor's latency, the online performance can reach 0.587 mAUC and 0.825 mDP. Taking the extra latency of ~ 5 ms/frame into account, the result will slightly suffer, decreasing to 0.576 mAUC and 0.807 mDP. Therefore, though PVT++ introduces extra latency, the online performance can still be significantly improved by more than **10**%.

K Training Set Analysis

Since PVT++ models can make full use of a trained tracker model, we find $\mathcal{P}_{\mathrm{V}}$ and $\mathcal{P}_{\mathrm{MV}}$ not very sensitive to the scale of training set. As shown in Table X, trained with only ~3,000 videos from VID (Russakovsky et al., 2015), our PVT++ can still converge well and achieve on par performance compared with the fully trained models.

L More Real-World Tests

In addition to the four real-world tests in Sec. 6.4 of the main paper, we present four more tests (together eight tests) in Fig. IV, where we implemented the models on a real UAV and performed several flight tests. The real-world tests involve two non-real-time trackers, SiamRPN++_Mob (Li et al., 2019) ($\sim$ 15.57 FPS in the tests) and SiamMask (Wang et al., 2019) ($\sim$ 11.95 FPS in the tests), which are largely affected by their high onboard latency. Coupled with our PVT++ ($\mathcal{P}_{\mathrm{MV}}$), the predictive models work well under various tracking scenes, *e.g.*, aspect ratio change in Test 1, dark environment in Test 2, 5, 7, and 8, view point change in Test 3, and occlusion in Test 2. The real-world tests also cover various target objects like person, building, car, and island, as shown in Fig. IV. The robustness of PVT++ in the onboard tests validate its effectiveness in the real-world UAV tracking challenges.