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 [47], UAV20L [47], DTB70 [37], and UAVDT [16], where the superiority of our PVT++ is clearly shown. In Appendix E, we present detailed results comparison between KF [30] and PVT++ to better demonstrate the superiority of our method. In addition to convolution neural network backbone[26, 53, 55]-based trackers, PVT++ further works for transformer-based ones [60, 11], which is presented in Appendix F. In Appendix G, we show that PVT++ is more efficiency and introduces much less extra latency onboard compared with other trajectory productors [24, 63, 40]. We also tried to fuse the motion and visual cues earlier in Appendix H, where we give an analysis to the strategy adopted in PVT++. The full attribute-based results from all the four datasets [47, 37, 16] are reported in Appendix I, 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 J, 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 K. We further find PVT++ is capable of converging well in smaller training set (using only 3563 videos from Imagenet VID [52]), which is shown in Appendix L. Finally, we present additional real-world tests in Appendix M, covering more target objects and tracking scenes.

B. Detailed Configuration

Specific Model Structure. Corresponding to Fig. 4 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.

Remark 5: These structures are general for all the four implemented base trackers [33, 57, 22].

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	C_{in}	C_{out}	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$_{\mathrm{t}}$	3×3	256	64	$B \times k \times 64 \times 29 \times 29$
	2D Conv$_{\mathrm{s}}$	3×3	256	64	$B \times k \times 64 \times 25 \times 25$
	2D Conv$_{\mathrm{e}}$	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$_{\mathrm{a}}$	1×1	64	64	$B \times k \times 64 \times 25 \times 25$
	2D Conv$_{\mathrm{a}}$	1×1	64	4	$B \times k \times 4 \times 25 \times 25$
	Avg. Pool$_{\mathrm{a}}$	-	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. 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 f_{j-1} to f_j	$\mathbb{R}^{1 \times 4}$
$\mathbf{p}_{f_j}$	Average moving speed from f_{j-k+1} to f_j	$\mathbb{R}^{1 \times 4}$
$\hat{\mathbf{m}}_f$	Predicted motion from $\phi(f)$ to f	$\mathbb{R}^{1 \times 4}$
$\mathbf{m}_f$	Ground-truth motion from $\phi(f)$ to f	$\mathbb{R}^{1 \times 4}$
Δ_f	Frame interval between the latest and the f-th frame	$\mathbb{R}$
$\Delta_{\hat{x}}(f), \Delta_{\hat{y}}(f)$	Predicted 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 $\phi(f)$	$\mathbb{R}^{C \times W \times H}$
$\mathbf{z}$	Template feature	$\mathbb{R}^{C \times a \times a}$
$k(=3)$	Number of past frames	$\mathbb{R}$
N	Number of the parallel FC layers in the decoder	$\mathbb{R}$

Training Settings. All the predictive modules need temporal video data for training. However, to our disappointment, existing training pipeline [33] 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 6: 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.

Table III. Per dataset results of different predictor modules. For all the three base trackers in various datasets, our PVT++ generally outperforms previous standard KF solutions [36, 32] and stronger learnable KF baselines, KF† and KF‡.

Dataset		DTB70		UAVDT		UAV20L		UAV123	
Tracker	Pred.	AUC@La0	DP@La0	AUC@La0	DP@La0	AUC@La0	DP@La0	AUC@La0	DP@La0
SiamRPN++$_{\mathrm{M}}$ (21FPS)	N/A	0.305	0.387	0.494	0.719	0.448	0.619	0.472	0.678
	KF [36]	0.349	0.482	0.527	0.737	0.458	0.624	0.515	0.712
	PVT [32]	0.377	0.518	0.533	0.740	0.458	0.624	0.522	0.722
	KF† [50]	0.367	0.504	0.519	0.732	0.466	0.630	0.511	0.703
	KF‡ [25]	0.365	0.496	0.563	0.780	0.483	0.658	0.513	0.598
	$\mathcal{P}_{\mathrm{M}}$ **(Ours)**	0.385	0.523	0.529	0.745	0.481	0.647	0.537	0.737
	$\mathcal{P}_{\mathrm{V}}$ **(Ours)**	0.352	0.472	0.564	0.799	0.488	0.675	0.504	0.703
	$\mathcal{P}_{\mathrm{MV}}$ **(Ours)**	**0.399**	**0.536**	**0.576**	**0.807**	**0.508**	**0.697**	**0.537**	**0.741**
SiamMask (12FPS)	N/A	0.247	0.313	0.455	0.703	0.405	0.571	0.436	0.639
	KF [36]	0.294	0.407	0.535	0.758	0.436	0.582	0.499	0.679
	PVT [32]	0.362	0.504	0.539	0.751	0.443	0.598	0.514	0.701
	KF† [50]	0.349	0.486	0.530	0.749	0.440	0.588	0.513	0.702
	KF‡ [25]	0.348	0.468	0.558	0.775	0.465	0.629	0.502	0.683
	$\mathcal{P}_{\mathrm{M}}$ **(Ours)**	**0.370**	**0.508**	0.531	0.760	0.449	0.607	0.532	0.743
	$\mathcal{P}_{\mathrm{V}}$ **(Ours)**	0.292	0.405	0.532	0.777	0.430	0.601	0.503	0.705
	$\mathcal{P}_{\mathrm{MV}}$ **(Ours)**	0.342	0.463	**0.566**	**0.797**	**0.469**	**0.644**	**0.536**	**0.749**
SiamRPN++$_{\mathrm{M}}$ (21FPS)	N/A	0.136	0.159	0.351	0.594	0.310	0.434	0.349	0.505
	KF [36]	0.189	0.232	0.451	0.667	0.387	0.528	0.415	0.582
	PVT [32]	0.201	0.254	0.467	0.687	0.396	0.547	0.434	0.605
	KF† [50]	0.200	0.254	0.460	0.680	0.412	0.572	0.433	0.603
	KF‡ [25]	0.204	0.252	0.504	0.728	0.406	0.549	0.432	0.599
	$\mathcal{P}_{\mathrm{M}}$ **(Ours)**	0.199	**0.258**	0.449	0.684	0.404	0.560	**0.442**	**0.627**
	$\mathcal{P}_{\mathrm{V}}$ **(Ours)**	0.179	0.225	0.403	0.665	0.398	0.548	0.398	0.559
	$\mathcal{P}_{\mathrm{MV}}$ **(Ours)**	**0.205**	0.256	**0.488**	**0.726**	**0.416**	0.568	0.442	0.619

All the PVT++ models are optimized by AdamW [45]. 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 four base trackers [33, 57, 22], $\mathcal{P}_{\mathrm{V}}$ 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^{-5}.

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++$_{\mathrm{M}}$ [33], $N = 12, \Delta_f = [1 : 12]$ for SiamRPN++$_{\mathrm{R}}$ [33], $N = 6, \Delta_f = [1 : 6]$ for SiamMask [57], and $N = 4, \Delta_f = [1 : 3]$ for SiamGAT [22]. 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.

Table IV. Efficiency and complexity comparison between PVT++ and other motion predictors [24, 63, 40]. Our framework is 10x∼100x faster than other works.

Input	Traj.			Traj. + RGB	
Model	Social GAN [24]	SR-LSTM [63]	$\mathcal{P}_{\mathrm{M}}$	NEXT [40]	$\mathcal{P}_{\mathrm{MV}}$
MACs	5.6M	51.7M	**0.05M**	2.7G	**1.2G**
Latency (ms)	50.2	652.0	**4.2**	181.6	**8.6**

Table V. Effect of PVT++ on transformer-based trackers [60, 11]. Our framework can boost the perfromance by up to **40**%.

Dataset	DTB70				UAVDT			
Metric	AUC@La0		DP@La0		AUC@La0		DP@La0	
PVT++	✗	✓	✗	✓	✗	✓	✗	✓
OSTrack [60]	0.306	**0.400**	0.375	**0.535**	0.533	**0.626**	0.789	**0.839**
MixFormer [11]	0.198	**0.250**	0.242	**0.320**	0.413	**0.516**	0.644	**0.719**

C. Complete Notation Reference Table

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

D. Visualization

We present some typical tracking visualization in Fig. I. The sequences, *ManRunning2*, *Paragliding5*, *Wakeboarding1*, and *Wakeboarding2* are from DTB70 [37].*S0303*, *S0304*, *S0310*, and *S1604* are from UAVDT [16]. In

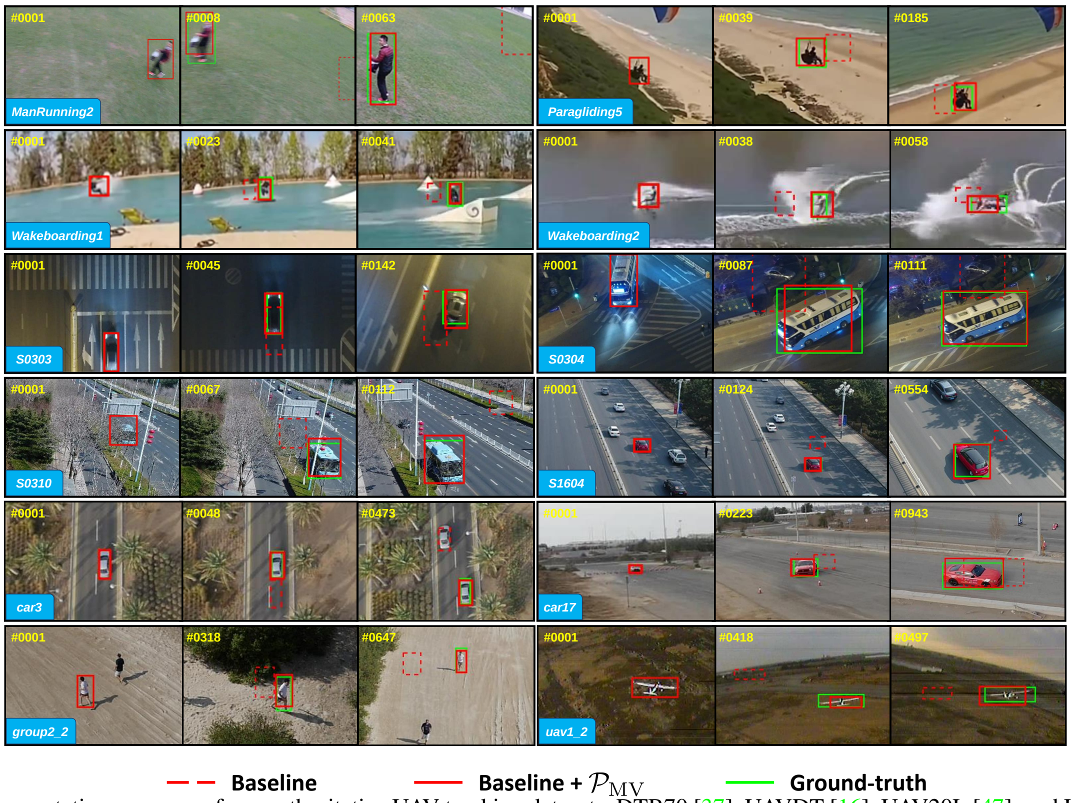

Figure I. Representative sequences from authoritative UAV tracking datasets, DTB70 [37], UAVDT [16], UAV20L [47], and UAV123 [47]. We use dashed red lines to demonstrate the original trackers, which are severely affected by onboard latency. Coupled with our PVT++ ($\mathcal{P}_{\rm MV}$), the robustness 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 VI. Results comparison between two fusion strategy. $\mathcal{P}_{\rm MV}$ denotes our default PVT++, the modalities fuse after independent temporal interaction (late fusion). $\mathcal{P}^{\dagger}_{\rm MV}$ 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
$\mathcal{P}_{\rm MV}$ (late fuse)	**0.399**	**0.536**	**0.576**	**0.807**
$\mathcal{P}^{\dagger}_{\rm MV}$ (early fuse)	0.370	0.498	0.571	0.800

UAV20L and UAV123 [47], 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 with our PVT++ ($\mathcal{P}_{\rm MV}$), the models (solid red boxes) are much more robust. We use greed boxes to denote ground-truth.

E. Prediction Quantitative Comparison

To provide a thorough quantitative comparison of the predictor performance, we reported the results per dataset in Table III. We observe that for different tracker models in various benchmarks, PVT++ is more robust than prior solutions [32, 36]. Compared with learnable KFs, KF† and KF‡, our PVT++ holds obvious advantage by virtue of the visual cue and joint learning.

F. Effect on Transformer-based Trackers

For transformer-based trackers, MixFormer [11] and OSTrack [60] ($\sim$6 and $\sim$10 FPS onboard), PVT++ yields up to 40% improvement as shown in Table V.

G. Efficiency and Complexity Comparison

PVT++ is a lightweight plug-and-play framework designed for latency-aware tracking, while most existing trajectory predictors are computationally heavy. As in Table IV, PVT++ is 10x$\sim$100x faster than existing trajectory predictors and introduces much less extra latency onboard.

H. 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*,

Table VII. Attribute-based analysis of the three trackers with PVT++ models in DTB70 [37] dataset.

Tracker		SiamRPN++$_{\mathrm{M}}$ (21FPS)				SiamRPN++$_{\mathrm{R}}$ (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	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

Table VIII. Attribute-based analysis of the three trackers with PVT++ models in UAVDT [16] dataset.

Tracker		SiamRPN++$_{\mathrm{M}}$ (21FPS)				SiamRPN++$_{\mathrm{R}}$ (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

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.

I. Full Attribute-based Analysis

We present full attribute-based analysis in Table VII, Table VIII, Table IX, and Table X. Following the previous work [37], we report results on aspect ratio variation (ARV), background 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 VII. As shown in Table VIII, 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 [16]. For UAV20L and UAV123 [47], 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 IX and Table X.

We observe that the two modalities has their own advantage in different UAV tracking challenges. For example, consider UAVDT dataset [16] (Table VIII), the visual

Table IX. Attribute-based analysis of the three trackers with PVT++ models in UAV20L [47] dataset.

Tracker		SiamRPN++$_{\mathrm{M}}$ (21FPS)				SiamRPN++$_{\mathrm{R}}$ (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 X. Attribute-based analysis of the three trackers with PVT++ models in UAV123 [47] dataset.

Tracker		SiamRPN++$_{\mathrm{M}}$ (21FPS)				SiamRPN++$_{\mathrm{R}}$ (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

branch is relatively good at challenges like camera rotation (CR), object rotation (OR), and scale variation (SV), where the object motion could be very complex and the visual appearance is helpful in prediction. On the other hand, motion cues are robust when the visual feature is not reliable, for instance, similar object (SO) and large occlusion (LO) challenge. In general, motion predictor is better than visual predictor, from which we conclude that past motion is still the main cue to inference future motion. While for the challenging UAV tracking, where motion could be extremely random and dynamic, introducing visual cues can significant improve the prediction robustness. Together, the

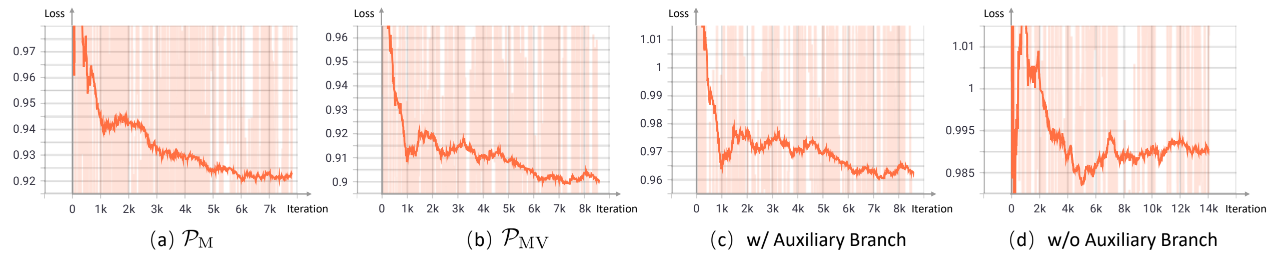

Figure II. Training loss curves of PVT++ models. Coupled with visual feature, $\mathcal{P}_{\mathrm{MV}}$ can better learn to predict than $\mathcal{P}_{\mathrm{M}}$, thus the loss is observed to be smaller. Without auxiliary branch, the loss curve is less smooth, indicating the importance of $\mathcal{A}$.

Table XI. Effect of extra latency brought by PVT++ in UAVDT [16] dataset. Here, the base tracker takes SiamRPN++$_{\mathrm{M}}$, whose original latency is fixed to 44.5 ms/frame (its average onboard latency). We use $\cdot^{\dagger}$ to indicate neglecting the latency. With ∼5ms/frame extra time, the performance is slightly lower (2∼3% performance drop), while it is acceptable and still brings upto 15% performance gain.

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 XII. Performance of PVT++ models trained with different datasets. Full denotes ∼9,000 videos from VID [52], LaSOT [18], and GOT-10k [28]. VID indicates using only ∼3,000 videos from VID [52]. 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

jointly optimized model $\mathcal{P}_{\mathrm{MV}}$ is the most reliable for UAV latency-aware vsiaul tracking.

J. Training Visualization

The training loss curves of PVT++ models with SiamRPN++$_{\mathrm{M}}$ [33] is shown in Fig. II. 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 fluctuates a lot, indicating that the model can't converge very well. We also noted that in terms of extra latency, PVT++ can achieve similar or smaller negative results compared to KFs. We assume this is because larger matrix operation can be more effectively realized on GPU, compared with small matrix on CPU/GPU.

K. Effect of Extra Latency

PVT++ will bring a bit extra latency during online perception, which is negative for the performance. As shown in Table XI, the latency of original tracker [33] is about 45 ms/frame. Ignoring 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**%.

L. 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 XII, trained with only ∼3,000 videos from VID [52], our PVT++ can still converge well and achieve on par performance compared with the fully trained models.

M. More Real-World Tests

In addition to the four real-world tests in Sec. 6.5 of the main paper, we present six more tests (together eight tests) in Fig. III, where we implemented the models on a real UAV and performed several flights. The real-world tests involve two non-real-time trackers, SiamRPN++$_{\mathrm{M}}$ [33] (∼ 15.57 FPS in the tests) and SiamMask [57] (∼ 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

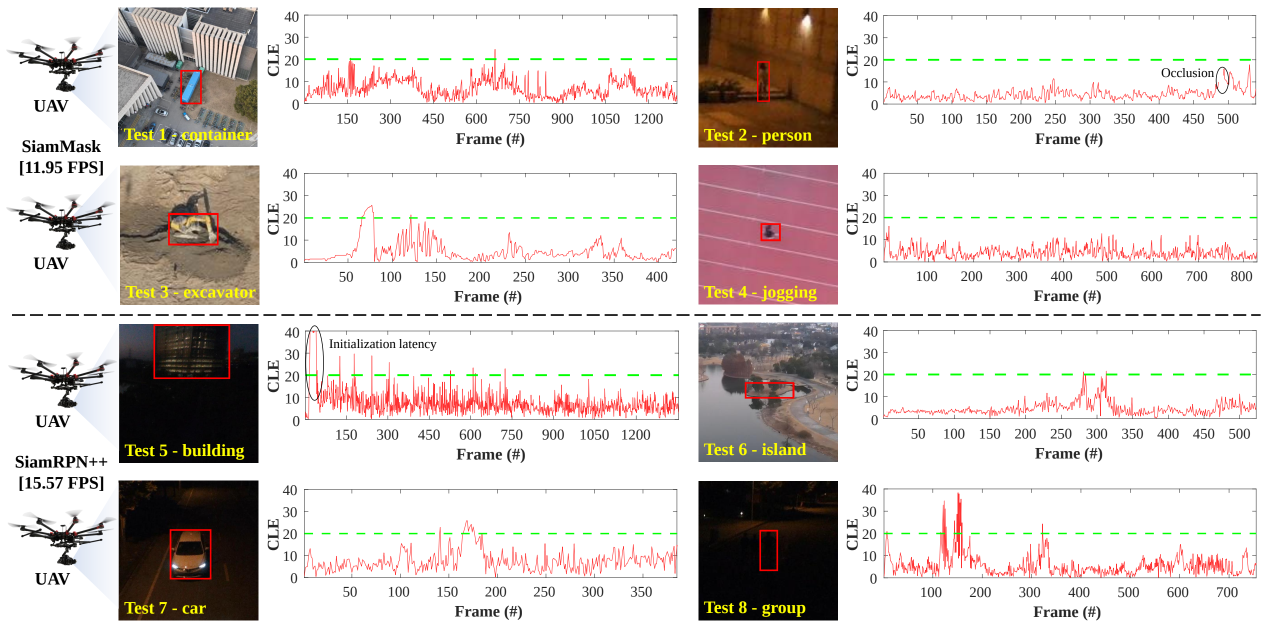

Figure III. Eight real-world tests of PVT++ on non-real-time trackers, SiamMask [57] and SiamRPN++$_{\mathrm{M}}$ [33]. We present the tracking scenes, the target objects, and center location error (CLE) in the figure. Under various challenges like aspect ration change, illumination variation, low resolution, PVT++ maintains its robustness, with CLE below 20 pixels in most frames.

person, building, car, and island, as shown in Fig. III. We have made them into videos for clear reference. The robustness of PVT++ in the onboard tests validate its effectiveness in the real-world UAV tracking challenges.