2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

CLAT: Convolutional Local Attention Tracker for Real-time UAV Target Tracking System with Feedback Information

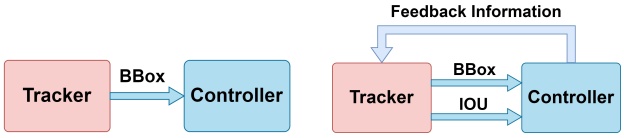
XiaoLou Sun*,[1,2], ZhiBin Quan*,[2], Wei Wang[3], WuFei Si[1], ChunYan Wang[1], YunTian Li[1], Yuan Wu⋆,[1], and Meng Shen⋆[1]

***Abstract*— Real-time UAV vision target tracking systems encounter the intricate challenges of striking a trade-off for tracking speed and performance, and the robustness of the following control. In existing tracking systems, the global attention mechanism enhances tracking performance, but it introduces higher computational complexity, impacting target tracking speed; the local attention mechanism can reduce computational complexity but often exhibits limitations in modeling the receptive field. In this paper, we propose a new framework named Convolutional Local Attention Tracker (CLAT) to address these challenges. Firstly, we design a hierarchical convolutional local attention structure as the feature extractor for CLAT. This leverages convolutional projection before local window partitioning, facilitating connections between non-overlapping windows and expanding the receptive field. Secondly, we introduce a streamlined feature fusion network comprising the unshared-weights convolutional layer and a global attention network. The whole design can balance speed and accuracy. Furthermore, to enhance servo control robustness, we have redesigned the upper-level controller by integrating all bounding box information. To capture feedback spatiotemporal information in CLAT, a dynamic template update is implemented by incorporating an IOU head into the predictor. Extensive experiments on visual tracking benchmarks and in the real world demonstrate that CLAT achieves competitive performance. Moreover, we have developed a comprehensive tracking system demonstration capable of precisely tracking targets across various categories. The tracker code will be released on `https://github.com/xiaolousun/refine-pytracking.git`.**

I. Introduction

Visual target tracking aims to track the specified target frame by frame, which is increasingly prevalent in unmanned aerial vehicle (UAV) applications, including obstacle avoidance, wildlife monitoring and target tracking, especially with the rise of intelligent devices. Balancing the accuracy and speed of the tracker deployed on embedded devices and maintain the servo robustness of the UAV is still a challenge, because the tracked target is often affected by significant changes in illumination, scale, background interference and serious obstacles, and the appearance of targets may also change in some cases[1, 2].

In recent years, siamese interaction-based algorithms have gained popularity for addressing tracking challenges through a combination of template matching and the acquisition of a generalized similarity feature map[1–3]. The style of the interaction is usually the correlation. But, the correlation operation entails a localized linear matching procedure, resulting in the depletion of semantic information for the tracker[4], and similarity maps with linear matching typically contain limited spatial information. Thanks to the application of the Transformer in visual tasks, global attention is also applied to the feature interaction between the template and search region to replace the correlation operation[4, 5]. It makes the performance of trackers scale new heights. Still, the design of the feature interaction based on global attention is typically complex and results in greater computational complexity stacked with more interaction layers to get better performance. Although this design achieves better results with the simplified architecture, the number of parameters and the complexity of the calculation far exceed all previous methods and fails to satisfy real-time constraints.



(a) Existing system pipeline (b) Our system pipeline

Fig. 1: The comparison between existing tracking system pipeline based on one-way circulation of visual information (a) and our tracking system pipeline based on the feedback information using IOU score (b).

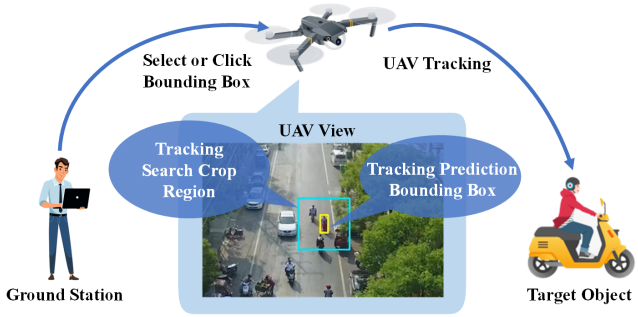


Fig. 2: The schematic diagram depicting the entire system: The initial bounding box can be manually delineated by human intervention or detector. Subsequently, it is transmitted to the UAV, which, upon receiving the initial bounding box, promptly initiates the tracking process of the target object.

To address the aforementioned challenges, we present a new framework named Convolutional Local Attention Tracker (CLAT) that can perform better than most trackers with global attention in visual tracking to balance performance and speed. We partition the tracking architecture into two distinct components: the Convolutional Local Attention backbone (CLANet) and the Convolutional Global Attention integration network (CGANet). Firstly, We design the shared-

⋆Corresponding Author and *The authors have the same contribution
[1]XiaoLou Sun, WuFei Si, ChunYan Wang, YunTian Li, Yuan Wu Meng Shen are with Purple Mountain Laboratories, Nanjing, China
[2]XiaoLou Sun, and ZhiBin Quan are with School of Automation, Southeast University, Nanjing, China
[3]Wei Wang is with School of Automation, Nanjing University of Information Science and Technology, Nanjing, China

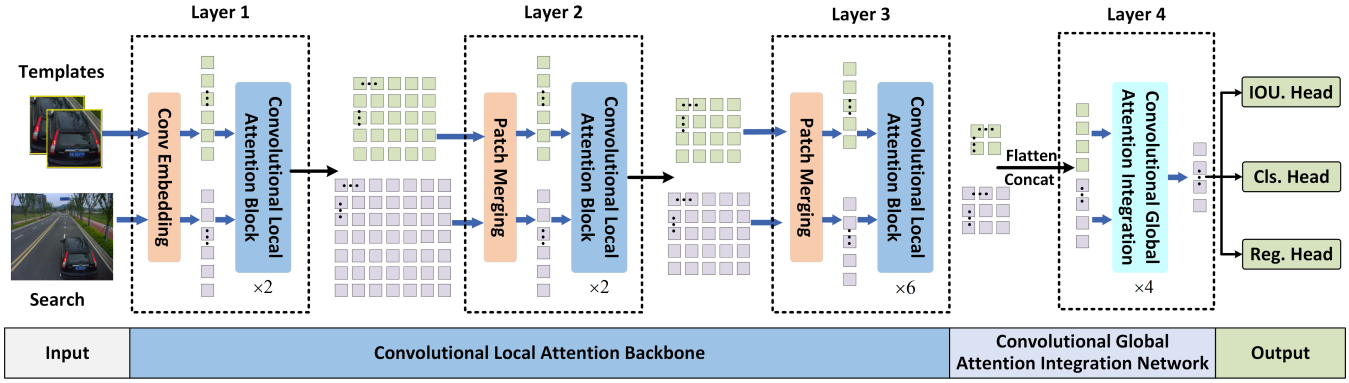


Fig. 3: Architecture of CLAT: The architecture is an end-to-end pyramid structure containing CLANet (layers 1-3) for feature extraction from templates and search regions, and CGANet (layer 4), which is used to fuse features.

weights hierarchy of the convolutional local attention as the feature extractor of the CLAT, leveraging the convolutional projection before the local windows partition to establish the connection between non-overlapped windows and enlarge the receptive field but also inherit the best of local attention to balance the speed and accuracy. Secondly, we propose a concise and practical feature fusion network composed of the unshared-weights convolution layer and global attention network. The extracted features through the unshared-weights convolution layer can be directly concatenated into the global attention network to obtain the fused features for classification and regression. Moreover, We exclusively employ the window penalty for post-processing purposes, removing all the other post-processing parameters to avoid parameter-sensitive situations for different scenes.

In addition to redesigning the visual tracking algorithm, we also develop an upper PD control algorithm that utilizes complete bounding box information, encompassing the parameters $[x, y, w, h]$, derived from visual tracking. The dynamic template update is used to get spatiotemporal information in CLAT by adding the IOU head to the tracker, as shown in Fig. 1(a), Fig. 1(b). The entire tracking process is illustrated in Fig 2. Primary contributions include:

- We introduce the Convolutional Local Attention Tracker (CLAT) for the task of visual target tracking consisting of the Convolutional Local Attention backbone (CLANet) and the Convolutional Global Attention integration network (CGANet). CLAT can significantly enlarge the receptive field and bolster the limited modeling capacity of local self-attention.
- The dynamic template feedback information update enables the entire tracking system to fully capture and utilize real-time long-time tracking information, making the whole UAV tracking system robust.
- The evaluations across multiple demanding benchmarks substantiate the exceptional performance of CLAT, compared to generic trackers and UAV-specialized trackers Furthermore, tests conducted on an embedded platform in the real world compellingly showcase the impressive practicality and real-time performance.

II. RELATED WORK

We provide a concise overview of two key areas closely related to our research: visual target tracking and window-based vision transformers.

A. Visual Target Tracking

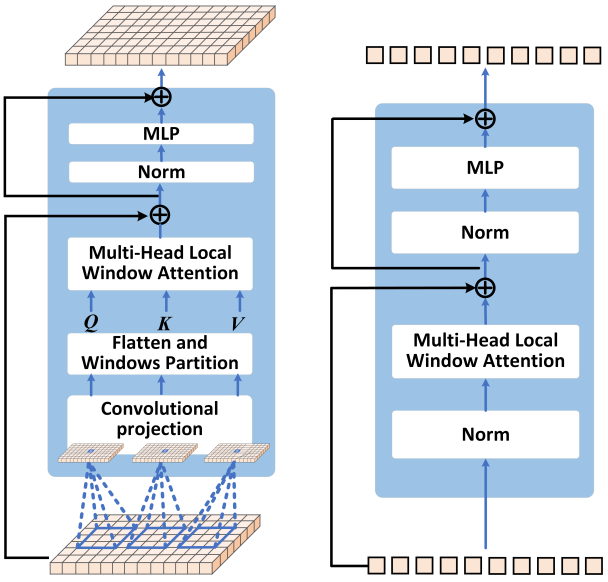
Accurate tracking is a subfield within computer vision that focuses on following targets as they traverse successive video frames, utilizing the initial knowledge provided by the target's appearance in the first frame as a reference.

The siamese network has garnered considerable interest in the past due to its outstanding performance and considerable potential for development. In recent years, with the development of the Transformer in vision, the trackers based on the global attention integration module like TransT[4], Stark[5] are presented to solve that the operation of linear correlation loses the semantic information. But, the design of the feature interaction based on global attention is typically complex and results in greater computational complexity by stacking more interaction layers to get better performance. Extral some trackers like SimTrack[6] use global attention as the backbone to directly fuse features from template tokens and search tokens through patch embedding. The final features generated by the network are utilized directly for classification and regression. Although this design achieves better results with the simplified architecture, the number of parameters and the complexity of the calculation far exceed all previous methods and can not meet the real-time requirements.

Inspired by these works, CLAT is designed considering the computational complexity and the design complexity of the backbone network and the interaction modules. We design the shared-weights hierarchy of the convolutional local attention as the feature extractor of the CLAT to balance the computational complexity and a concise and practical feature fusion network composed of the unshared-weights convolution layer and global attention network to balance the design complexity of the interaction modules.

B. Vision Transformers Based On Window Attention

While global Vision Transformers have achieved success in image classification, greater challenges encounter when



(a) Convolutional local attention (b) Typical local attention

Fig. 4: Illustration of our convolutional local attention (a) and the typical local attention (b).

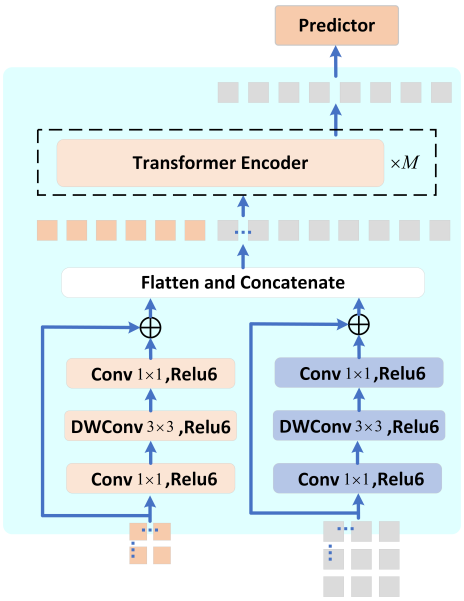


Fig. 5: Illustration of our Convolutional Global Attention Integration Network (CGANet). The whole architecture is composed of the unshared-weights convolution layer and the transformer network (global attention).

applied to downstream tasks, especially in high-resolution vision applications. The computational cost of Vision Transformers escalates quadratically with image size, rendering it impractical for real-world applications. Various methods have been proposed recently to make vision transformers more versatile as general backbones for downstream tasks. One such method is the Window-based Vision Transformer (WBVT)[7], which employs a local window attention mechanism that increases computational complexity linearly with image size. However, this approach limits the receptive fields in local windows, which are critical for downstream vision tasks. Researchers have proposed several techniques to connect nearby windows, including shifting[8], cross shaped[9], shuffling operations[7]. In addition, some researchers employ convolutions to enhance the receptive fields, as convolution layers naturally capture local relations.

On the shoulders of the aforementioned work, we design the shared-weights hierarchy of the convolutional local attention as the feature extractor of the CLAT, leveraging the convolutional projection before the local windows partition to establish the connection between non-overlapped windows and enlarge the receptive field.

III. CLAT FRAMEWORK

In this section, we describe our CLAT framework in detail. As shown in Fig. 3, CLAT is designed as an end-to-end pyramid structure containing CLANet (layers 1-3, CLAT) and CGANet (layer 4, CLAT). Table I shows the series.

A. Convolutional Local Attention Backbone

1) Convolutional Local Attention block: Attention, including local attention and global attention, is the fundamental component of CLAT. When provided with queries Q, keys K, and values V, the attention function employed is the scaled dot-product, as follows,

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}, \quad (1)$$

where d_k is the key dimensionality.

As previously discussed, expanding the attention mechanism into multiple heads allows for various attention distributions to be considered, enabling the model to focus on different aspects of information. The formulation for the multi-head attention mechanism is presented in equation (2)-(3). For a more comprehensive explanation, we direct interested readers to relevant literature[10].

$$\mathbf{H}_i = \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right), \quad (2)$$

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}\left(\mathbf{H}_1,\ldots,\mathbf{H}_{n_h}\right)\mathbf{W}^O, \quad (3)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_k}$, and $\mathbf{W}^O \in \mathbb{R}^{n_h d_v \times d_m}$, are parameter matrices. In this work, we employ $n_h = [3,6,12,24]$, $d_m = [96,192,384]$, and $d_k = d_v = d_m/n_h = 32$ as default values.

The entire process of convolutional projection is shown in Fig. 4. Tokens are initially transformed into a 2D token map. Following that, A convolutional projection is employed by utilizing a convolutional layer equipped with a kernel of size s. Finally, the projected tokens are passed through window partitions and flattened into a 1D form for further processing. This process can be expressed as follows:

$$x_i^q = \text{Flatten}(\text{Partition}(\text{Conv}(\text{Reshape}(x_i), s))), \quad (4)$$

where $x_i^{q/k/v}$ represents the token input pertaining to the $Q/K/V$ matrices at layer i, while x_i denotes the original, unmodified token prior. In the case of the window partitioning technique, the successive CLAT blocks are computed as:

$$\hat{\mathbf{x}}^l = \text{C-WMSA}\left(\text{LN}\left(\mathbf{x}^{l-1}\right)\right) + \mathbf{x}^{l-1}, \quad (5)$$

$$\mathbf{x}^l = \text{FFN}\left(\text{LN}\left(\hat{\mathbf{x}}^l\right)\right) + \hat{\mathbf{x}}^l, \quad (6)$$

where C-WMSA represents convolutional window-based self attention, LN represents layer norm, and FFN represents feed forward network.

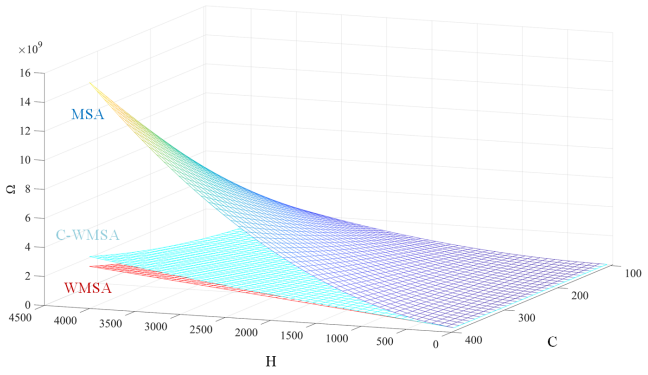


Fig. 6: Illustration of the computational complexity. The computational complexity of CLANet is not completely linear, but it decreases exponentially compared to global attention.

Assuming that each window comprises $M \times M$ patches, we can estimate the computational complexity of a multi-head self-attention module, a window-based module, and a convolutional window-based module on images of $H(H = h \times w)$ patches are as follows and Fig. 6 shows the computational complexity surface,

$$\Omega(\text{MSA}) = 4HC^2 + 2(H)^2C, \tag{7}$$

$$\Omega(\text{W-MSA}) = 4HC^2 + 2M^2HC, \tag{8}$$

$$\Omega(\text{C-WMSA}) = 4HC^2 + 2M^2HC + 3(\frac{H}{M})^2 s^2 d. \tag{9}$$

where s represents the size of the convolution kernel, d represents the number of convolution kernels.

B. Convolutional Global Attention Integration Network

As depicted in Fig. 5, A detailed description of the CGANet utilized in CLAT is provided. The feature tokens extracted by the CLANet are processed through unshared-weight convolutional layers, which enable the network to emphasize specific features of the template or search region. Then, unlike the baseline model, we directly concatenate z^0 and x^0, which are the feature tokens through the convolution of the template and search, and send them to the Transformer encoder network together to fuse the features for classification and regression,

$$\begin{bmatrix} z^* \\ x^* \end{bmatrix} = \begin{bmatrix} z^l \\ x^l \end{bmatrix} + Att\left(\begin{bmatrix} z^l \\ x^l \end{bmatrix}\right), \tag{10}$$

$$\begin{bmatrix} z^{l+1} \\ x^{l+1} \end{bmatrix} = \begin{bmatrix} z^* \\ x^* \end{bmatrix} + FFN\left(\begin{bmatrix} z^* \\ x^* \end{bmatrix}\right). \tag{11}$$

The symbol for layer normalization is omitted for simplicity. The primary distinction between equation (1) and equation (10) lies in the computation of $\text{Att}(\cdot)$,

$$\begin{cases} M_{z^l}(z^l, x^l) = \left[\ a\left(z^l, z^l\right), a\left(z^l, x^l\right)\ \right], \\ M_{x^l}(z^l, x^l) = \left[\ a\left(x^l, z^l\right), a\left(x^l, x^l\right)\ \right] \end{cases} \tag{12}$$

$$\text{SM}\left(\begin{bmatrix} z^l \\ x^l \end{bmatrix}\right) = \text{Softmax}\left(\begin{bmatrix} M_{z^l}(z^l, x^l) \\ M_{x^l}(z^l, x^l) \end{bmatrix}\right), \tag{13}$$

$$\text{Att}\left(\begin{bmatrix} z^l \\ x^l \end{bmatrix}\right) = \text{SM}\left(\begin{bmatrix} z^l \\ x^l \end{bmatrix}\right)\left(\begin{bmatrix} z^l W_V \\ x^l W_V \end{bmatrix}\right), \tag{14}$$

where $a(x, y) = (xW_Q)(yW_K)^T/\sqrt{d}$. After converting equation (12)-(14), the template attention $\text{Att}(z^l)$ and search attention $\text{Att}(x^l)$ are:

$$\text{Att}\left(z^l\right) = \text{Softmax}\left(M_{z^l}(z^l, x^l)\right)\left[z^l W_V, x^l W_V\right]^T, \tag{15}$$

$$\text{Att}\left(x^l\right) = \text{Softmax}\left(M_{x^l}(z^l, x^l)\right)\left[z^l W_V, x^l W_V\right]^T. \tag{16}$$

The learning features of templates and search tokens mutually influence each other through $a(s^l, z^l)$ and $a(z^l, s^l)$. The component $\text{Att}(z^l)$ encompasses information derived from x^l, and conversely, x^l incorporates information from $\text{Att}(z^l)$. This reciprocal information exchange among tokens from both the template and search regions is a consistent feature throughout each layer of our integration network. Therefore, adding an extra interaction module is unnecessary. We forward the output search feature tokens x^l to the predictor for localization. The visualization results of each layer for CGANet are depicted in Fig. 7. The results show the features can shield other interference perfectly.

TABLE I: Elaborate settings of CLAT's architecture variants. To ensure a fair comparison and substantiate whether our proposed method addresses the issue of local attention raised within Swin[8], we employed layer numbers equal to Swin.

Method	Conv Type	Kernel Szie
CLAT-Tiny-dw3	depthwise convolution	3×3
CLAT-Tiny-dw5	depthwise convolution	5×5
CLAT-Tiny-dw7	depthwise convolution	7×7
CLAT-Tiny-df3	deformable convolution	3×3
CLAT-Tiny-dl3	dilated convolution	3×3

IV. UAV TRACKING SYSTEM BASED ON CLAT

In this section, we provide a comprehensive description of our UAV tracking system and delve into the detailed design of the control algorithm.

A. Control Algorithm Design

We employ the PD control algorithm[11] for following control which leverages the target's center point shifts and integrates the changes in bounding box scale to calculate the reference velocity. One of the most critical steps before following is to map the quantities on the pixel coordinate system to the camera coordinate system, which can be achieved by the image Jacobi matrix, as follows:

$$\begin{cases} \bar{u} = u - u_{\text{center}}, \\ \bar{v} = v - v_{\text{center}}, \end{cases} \tag{17}$$

$$M(\dot{\bar{u}}, \dot{\bar{v}}) = \begin{bmatrix} -\frac{f_x}{z} & 0 & \frac{\bar{u}}{z} & \frac{\bar{u}\bar{v}}{f_x} & -\frac{f_x^2+\bar{u}^2}{f_x} & \bar{v} \\ 0 & -\frac{f_y}{z} & \frac{\bar{v}}{z} & \frac{f_y^2+\bar{u}^2}{f_y} & \frac{\bar{u}\bar{v}}{f_y} & -\bar{u} \end{bmatrix}, \tag{18}$$

$$\begin{bmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \end{bmatrix} = M(\bar{u}, \bar{v})\begin{bmatrix} T_x & T_y & T_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T, \tag{19}$$

where $\bar{u}, \bar{v}$ is the pixel coordinate difference, f_x, f_y is the equivalent focal length of the camera internal reference, z is the camera mapping depth, $\dot{\bar{u}}, \dot{\bar{v}}$ is the reference velocity, $T_x, T_y, T_z, \omega_x, \omega_y, \omega_z$ is the linear velocities and angular velocities of the three axes in the camera coordinate system.

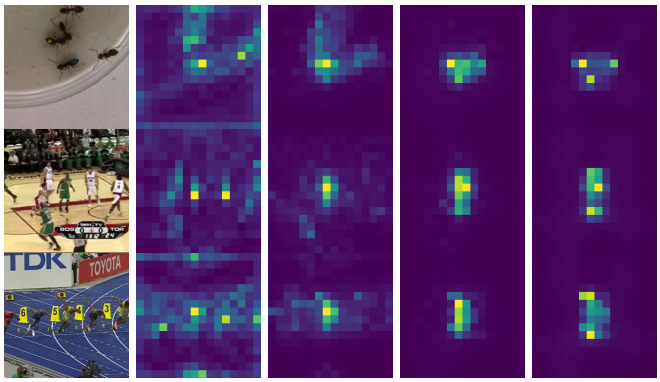


Fig. 7: The visualization results of the each layer (from left to right, corresponding to layers 1 to 4) for CGANet layer 4 interior. The results show the features can shield other interference perfectly.

In practical application, our PD control algorithm is designed to take into account the center point drift and bounding box information. Meanwhile, the IOU score is introduced as the confidence of the real-time bbox,

$$\dot{\bar{u}} = -k_p \bar{u} + \frac{k_d}{T}(u_i - u_{i-1}), \tag{20}$$

$$\dot{\bar{v}} = -k_p(\frac{w_0}{w_i} + \frac{h_0}{h_i})score \cdot \bar{v} + \frac{k_d}{T}(v_i - v_{i-1}), \tag{21}$$

where w_0, h_0 are the initial bbox from the ground station, w_i, h_i are the real-time bbox predicted from the tracker, the $score$ is the IOU score predicted by the tracker.

We designed the UAV tracking system by fixing the camera on the UAV while tilting it down θ degree. So, $\omega_x = 0, \omega_y = 0, \omega_z = 0$. In most cases, the UAV is flying at a fixed height during the tracking process. So, the linear velocity of the UAV in the Z-axis is 0, *i.e.*, $v_z = 0$.

We assume that the height of the tracked ground target is negligible in the UAV's view. So, $z = h/\sin\theta$, and we can obtain the following velocity transformation(v_x, v_y) equation between the camera coordinate system and the UAV coordinate system as follows:

$$\begin{cases} T_x = -v_y, \\ T_y = -v_x \sin\theta, \\ T_z = v_x \cos\theta. \end{cases} \tag{22}$$

Combining all the above conversion processes, the final control method can be summarized as follows:

$$v_x = \frac{\dot{\bar{v}} z}{\bar{v}\cos\theta + f_y \sin\theta}, \tag{23}$$

$$v_y = \frac{z\dot{\bar{u}} - \bar{u}\cos\theta v_x}{f_x}. \tag{24}$$

V. EXPERIMENTS

A. Implementation Details

1) CLAT Trainning: We have implemented CLAT in Python using PyTorch and utilized 8 Nvidia RTX 3090 GPUs for the computation. The way of training is the same as TransT[4]. It should be noted that we only use it in the last block of each layer to save computational overhead when we use deformable convolution and dilated convolution for convolution projection. To initially validate our approach, we conducted a classification experiment on the ImageNet-1K dataset[22]. CLANet yielded competitive results, as demonstrated, and Table II shows the details.

2) CLAT deploying: We deployed our CLAT on Nvidia Xavier Origin, encapsulating it into a ROS node. It is noteworthy that during the practical deployment of the UAV, our feedback information aligns with the frequency of the UAV flight controller rather than that of the Origin.

TABLE II: The classification accuracy on the ImageNet (using a single 224-pixel crop for assessing performance. The data is from the source official paper and github).

Method	Params	TOP-1	TOP-5
Conv Method			
ResNet-101[12]	45M	77.4	93.95
ConvMixer-768[13]	21M	80.2	95.08
ConvNext[14]	25M	81.6	95.89
RegNetY-8G[15]	39M	81.7	95.03
Vison Transformers			
PVT-L[16]	61M	81.7	/
CvT-21[17]	32M	82.5	/
TwinsP-S[18]	24M	81.2	95.69
Swin-T[8]	29M	81.3	95.5
Focal-T[19]	29M	82.2	95.9
Shuffle-T[7]	29M	82.5	/
Mixformer[20]	29M	82.5	/
CLANet-Tiny-dw3	28.43M	81.5	95.9
CLANet-Tiny-dw5	28.64M	81.5	95.57
CLANet-Tiny-dw7	28.96M	81.7	95.65
CLANet-Tiny-df3	34.13M	82.0	96.04
CLANet-Tiny-dl3	33.64M	81.6	95.7

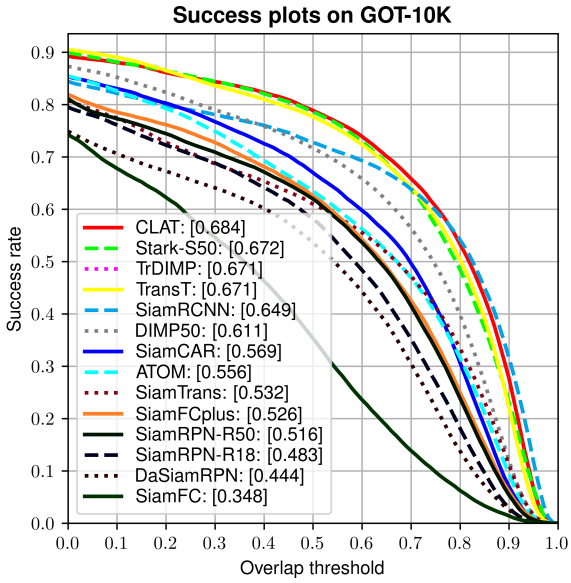


Fig. 8: Evaluations on GOT-10K[21]. CLAT exhibits remarkable performance, outperforming state-of-the-art trackers.

B. Results on GOT-10K

The GOT-10k[21] benchmark is a noteworthy benchmark in the field, offering a diverse and extensive collection

TABLE III: Comparisons on GOT-10K. The top two results are distinguished with blue and red fonts highlights.

Tracker	Backbone Net	Size	Net Type	AO	$SR_{0.5}$	$SR_{0.75}$	FPS	Params	FLOPs	Hardware
SiamRPN[23]	ResNet18	255	CNN	48.3	58.1	27	36.3	28.2M	12.8G	RTX 3060
SiamRPN++[3]	ResNet50	256	CNN	51.7	61.6	32.5	8.2	53.95M	48.92G	RTX 3060
DiMP[24]	ResNet50	288	CNN	61.1	71.1	49.2	42.4	43.10M	10.35G	RTX 3060
TransT[4]	ResNet50	256	CNN+Transformer	67.1	76.8	60.9	31.6	23.02M	16.71G	RTX 3060
STARK-S[5]	ResNet50	320	CNN+Transformer	67.2	76.1	61.2	36.3	28.2M	12.8G	RTX 3060
STARK-ST[5]	ResNet101	320	CNN+Transformer	68.8	78.1	64.1	24.8	47.2M	20.4G	RTX 3060
Swin-Track[2, 25]	Local Attn	384	Transformer	69.4	78.0	64.3	11.3	90.96M	61.85G	RTX 3060
SparseTT[26]	Local Attn.	256	Transformer	69.3	79.1	63.8	33.8	46.33M	15.08G	RTX 3060
SimTrack-B[6]	Global Att.	224	Transformer	68.6	78.9	62.4	43.3	88.64M	22.26G	RTX 3060
CLAT(Only GOT-10K)	Local Att.	256	Transformer	68.4	78.9	60.6	68.1	23.45M	10.36G	RTX 3060
CLAT(All Datasets)	Local Att.	256	Transformer	71.6	83.2	66.0	68.1	23.45M	10.36G	RTX 3060

TABLE IV: Comparisons on UAV123, DTB70. The top two results are distinguished with blue and red fonts highlights.

Tracker	Source	Net Type	Arm Real-time	UAV123		DTB70		UAV123@10fps	
				$Succ.$	$Prec.$	$Succ.$	$Prec.$	$Succ.$	$Prec.$
DaSiamRPN[27]	ECCV2018	CNN	✓	50.1	72.5	47.2	66.4	48.1	69.2
C-COT[28]	ECCV2016	Correlation Filters	✓	50.2	72.9	51.7	76.9	50.3	70.6
DeepSTRCF[29]	CVPR2018	CNN	✓	50.8	70.5	50.6	73.4	49.9	68.2
SiamAPN++[30]	IROS2021	CNN	✓	57.9	76.4	59.4	78.9	58	76.4
SiamAPN[31]	ICRA2021	CNN	✓	57.5	76.5	58.6	78.4	56.6	75.2
HiFT[32]	ICCV2021	CNN+Transformer	✓	58.9	78.7	59.4	80.2	74.9	56.9
TCTrack[33]	CVPR2022	CNN+Transformer	✓	60.4	80	62.2	81.3	58.8	77.4
SGDVIT[34]	ICRA2023	CNN+Transformer	✓	-	-	60.3	80.6	58.5	76.6
CLAT(ours)	-	Transformer	✓	66.1	85.9	67.8	88.3	63.6	81.9

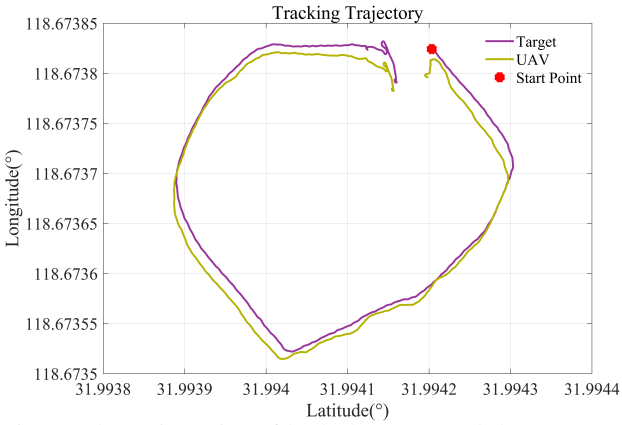


Fig. 9: The trajectories of both the UAV and the target, with position information acquired from GNSS data.

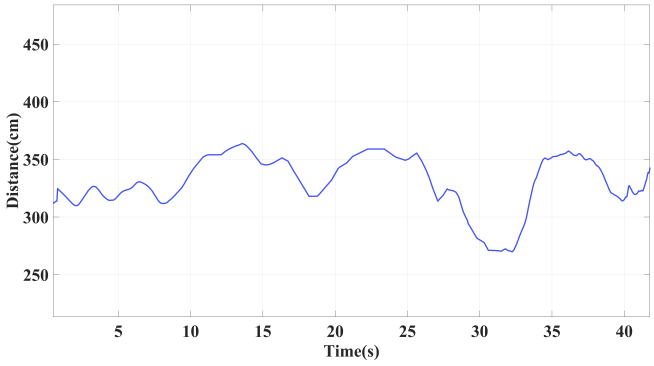


Fig. 10: The representation of the relative distance between the UAV and the target. Despite variations in the target's speed and direction, the relative distance consistently remains within a stable range.

TABLE V: Comparing the effect of CLAT with and without convolution and with different sizes of convolution kernels.

Pretrained	Convolution		GOT-10K		
	Size	Type	AO	$SR_{0.5}$	$SR_{0.75}$
-	-	-	65.2	74.9	55.6
✓	-	-	66.2	75.8	58.6
-	3×3	dw-bn	66.1	75.0	58.7
✓	3×3	dw-bn	67.1	77.7	57.7
-	5×5	dw-bn	66	76.6	56.1
-	7×7	dw-bn	65.2	75.3	55.6
✓	3×3	df-bn	67.2	78.2	57.8
✓	3×3	dl-bn	67.1	77.8	57.3

of targets in real-world scenarios. As depicted in Fig. 8, CLAT consistently demonstrates superior performance when compared to competitive generic trackers on the GOT-10K dataset. Detailed results are provided in Table III. CLAT with a reduced parameter and lower FLOPs, can showcase superior performance and be suitable for the arm devices.

TABLE VI: Comparative analysis of CLANet and Other local attention methods on tracker performance.

Pretrained	Window Connect	GOT-10K		
	Type	AO	$SR_{0.5}$	$SR_{0.75}$
✓	shuffle[7]	62.7	71.3	54.3
✓	cross shape[9]	64.5	74.6	56.2
✓	shift window [8]	63.8	72.6	55.8
✓	deformable bias[35]	66.4	77.0	59.6
✓	convolution project	67.5	78.9	58.9

C. Results on UAV123 and DTB70

UAV123 dataset [36], comprises a collection of 123 video sequences, encompassing over 110,000 frames including 30 FPS and 10 FPS. Additionally, DTB70 dataset[37], consists of 70 demanding UAV sequences, featuring numerous challenging motion scenes. Remarkably, CLAT demonstrates its superiority over other competitive UAV-specialized trackers when evaluated on both the UAV123 and DTB70 datasets achieving real-time speed in Arm, as outlined in Table IV.

D. Ablation Study

To evaluate the effectiveness of convolution within the proposed method, comprehensive studies comparing CLAT with various convolution were conducted and trained without any data augmentation. To validate the effectiveness of convolutional projection for object tracking tasks, we conducted a comparative analysis with other works that establish local window connections. In this comparison, we exclusively replaced the feature extraction component while keeping all other elements constant, and then proceeded with retraining the model and testing on GOT-10K.

We specifically build ablation experiments to verify the role and the effect of the size of the convolution kernel on the experiment. The results show the whole network is significantly enhanced in terms of metrics after the introduction of convolution according to the line 1-4 in Table V. It also shows that the larger convolution kernel is unsuitable for feature extraction in CLAT and, CLAT with pretrained deformable convolution can get the best results according to the line 3-8 in Table V. We constructed relevant comparative experiments to juxtapose our convolutional projection method against other local window correlation approaches, using the 224-pixel pretrained model (accordingly, template and search image are 112 and 224 pixel crop). The experimental results are presented in Table VI. As evident from the table, our method outperforms other approaches in tracking tasks. We offer the following explanation for this superior performance: Object tracking, due to its inherent complexity, typically requires backbone with a more robust long-range modeling capability. Pure physical operations tend to have weaker data adaptability and long-range modeling capacity. Our method, however, learns window modeling simultaneously during the projection of q, k, and v, making it more suitable for object tracking tasks. The effectiveness of the deformable bias method indirectly corroborates this assertion.

E. Results in Real World

We conducted an extensive outdoor tracking test to validate the convergence of our deployed unmanned aerial vehicle system. This involved tracking a target with significant motion over an extended duration, recording both its trajectory and the UAV's tracking path. Concurrently, we documented the relative distance, as illustrated in Fig. 9 and Fig. 10. The validation through trajectory and distance attests to the convergence of the tracking system.

VI. CONCLUSIONS

In this paper, we introduce a novel framework named Convolutional Local Attention Tracker (CLAT) to establish the connection between non-overlapped windows and enlarge the receptive field but also inherit the best of local attention to strike a balance between speed and performance. In order to bolster the servo control robustness, the upper control component has been developed to incorporate all bounding box information. The dynamic template update is used to get spatiotemporal information in CLAT by adding the IOU head to the predictor. Experimental results demonstrate that our designed tracking system consistently maintains the tracking performance steady and robust in both benchmark datasets and real-world scenarios.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62006041, 61773117, and in part by the Macao Young Scholars Program under Grant AM201914.

REFERENCES

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[2] J. Kugarajeevan, T. Kokul, A. Ramanan, and S. Fernando, "Transformers in single object tracking: an experimental survey," *IEEE Access*, 2023.

[3] B. Li, W. Wu, Q. Wang, F. Zhang, J. 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*, 2019, pp. 4282–4291.

[4] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8126–8135.

[5] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning spatio-temporal transformer for visual tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 448–10 457.

[6] B. Chen, P. Li, L. Bai, L. Qiao, Q. Shen, B. Li, W. Gan, W. Wu, and W. Ouyang, "Backbone is all your need: a simplified architecture for visual object tracking," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer, 2022, pp. 375–392.

[7] Z. Huang, Y. Ben, G. Luo, P. Cheng, G. Yu, and B. Fu, "Shuffle transformer: Rethinking spatial shuffle for vision transformer," *arXiv preprint arXiv:2106.03650*, 2021.

[8] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[9] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo, "Cswin transformer: A general vision transformer backbone with cross-shaped windows," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 124–12 134.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[11] X. Sun, Q. Wang, F. Xie, Z. Quan, W. Wang, H. Wang, Y. Yao, W. Yang, and S. Suzuki, "Siamese transformer network: Building an autonomous real-time

target tracking system for uav," *Journal of Systems Architecture*, vol. 130, p. 102675, 2022.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] A. Trockman and J. Z. Kolter, "Patches are all you need?" *arXiv preprint arXiv:2201.09792*, 2022.

[14] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 976–11 986.

[15] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 428–10 436.

[16] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 568–578.

[17] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 22–31.

[18] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting the design of spatial attention in vision transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9355–9366, 2021.

[19] J. Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, and J. Gao, "Focal self-attention for local-global interactions in vision transformers," *arXiv preprint arXiv:2107.00641*, 2021.

[20] Q. Chen, Q. Wu, J. Wang, Q. Hu, T. Hu, E. Ding, J. Cheng, and J. Wang, "Mixformer: Mixing features across windows and dimensions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5249–5259.

[21] L. Huang, X. 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*, 2019.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[23] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8971–8980.

[24] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6182–6191.

[25] L. Lin, H. Fan, Z. Zhang, Y. Xu, and H. Ling, "Swintrack: A simple and strong baseline for transformer tracking," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 743–16 754, 2022.

[26] Z. Fu, Z. Fu, Q. Liu, W. Cai, and Y. Wang, "Sparsett: Visual tracking with sparse transformers," *arXiv preprint arXiv:2205.03776*, 2022.

[27] Z. Zhu, Q. Wang, B. Li, W. 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, pp. 101–117.

[28] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *European conference on computer vision*. Springer, 2016, pp. 472–488.

[29] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 58–66.

[30] Z. Cao, C. Fu, J. Ye, B. Li, and Y. Li, "Siamapn++: Siamese attentional aggregation network for real-time uav tracking," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3086–3092.

[31] T. H. Dinh, L. T. Quoc, and K. T. Trung, "Siamese attention and point adaptive network for visual tracking," in *2021 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*. IEEE, 2021, pp. 1–6.

[32] Z. Cao, C. Fu, J. Ye, B. Li, and Y. Li, "Hift: Hierarchical feature transformer for aerial tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 457–15 466.

[33] Z. Cao, Z. Huang, L. Pan, S. Zhang, Z. Liu, and C. Fu, "Tctrack: Temporal contexts for aerial tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 798–14 808.

[34] L. Yao, C. Fu, S. Li, G. Zheng, and J. Ye, "Sgdvit: Saliency-guided dynamic vision transformer for uav tracking," *arXiv preprint arXiv:2303.04378*, 2023.

[35] Z. Xia, X. Pan, S. Song, L. E. Li, and G. Huang, "Vision transformer with deformable attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 4794–4803.

[36] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European conference on computer vision*. Springer, 2016, pp. 445–461.

[37] S. Li and D.-Y. Yeung, "Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.