Exploring Convex Optimization and Transformer based Methods for Efficient Visual Object Tracking

Goutam Yelluru Gopal

A Thesis
In The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

April 2024

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Goutam Yelluru Gopal**

Entitled: **Exploring Convex Optimization and Transformer based Methods for Efficient Visual Object Tracking**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Javad Dargahi _____ Chair

Dr. Robert Laganière _____ External Examiner

Dr. Charalambos Poullis _____ External to the Department

Dr. Hassan Rivaz _____ Examiner

Dr. Luis Rodrigues _____ Examiner

Dr. Maria Amer _____ Thesis Supervisor

Approved by: _____

Dr. Jun Cai, Graduate Program Director

April 9, 2024 _____

Dr. Mourad Debbabi

Dean

Gina Cody School of Engineering and Computer Science

Abstract

Exploring Convex Optimization and Transformer based Methods for Efficient Visual Object Tracking

Goutam Yelluru Gopal, Ph.D.
Concordia University, 2024

The effectiveness of a visual object tracking algorithm heavily relies on how well it represents the target object through a collection of feature templates, or channels. However, some channels lose their discriminative power during challenging video conditions such as target deformation, occlusion, and motion blur, leading to tracking failures. Discriminative Correlation Filter-based (DCF) trackers address these video challenges by aggregating hand-crafted and deep Convolutional Neural Network-based (CNN) channels. However, this approach increases the computational complexity of the tracker and significantly reduces inference speed, especially on constrained hardware such as a Central Processing Unit (CPU) or edge devices. We observe a parallel trend in end-to-end trainable deep Siamese Network-based (SN) trackers, which deploy parameter-heavy backbones for feature extraction and rely on specialized hardware such as a Graphics Processing Unit (GPU) for faster inference. In this thesis, we propose computationally efficient solutions to both DCF and SN tracking algorithms while improving their accuracy.

For multi-channel DCF tracking, we present three solutions to alleviate the impact of non-discriminative features (or channels). These methods leverage the concept of reliability to quantify the discriminative power of a feature (or a channel) based on its filter response. The proposed solutions dynamically lower the weightage of unreliable features (or channels) while emphasizing the temporal smoothness of the learned weights. We formulate the process of learning adaptive weights as a convex optimization problem and derive efficient solutions to maintain tracking speed. Expanding on the lightweight SN tracking paradigm, our first algorithm, *MVT*, employs a cascaded arrangement of CNN and transformer blocks in its backbone. This approach fuses template and search regions during feature extraction to generate superior feature encoding for target localization. Our second tracking algorithm, a Separable Self and Mixed Attention Transformer-based tracker (*SMAT*), further increases the efficiency of *MVT* by replacing the standard attention with a computationally efficient

separable attention block. Proposed trackers exhibit superior performance on eight challenging benchmarks compared to the related lightweight trackers, with *SMAT* emerging as the top performer. The computationally efficient architecture enables our *MVT* and *SMAT* trackers to run at real-time tracking speed on a CPU, while achieving a high speed of 150 frames-per-second on a GPU.

Acknowledgments

I would like to express my heartfelt gratitude to all those who have played a significant role in the completion of this challenging thesis journey, particularly during the testing times of the COVID pandemic. The unwavering support, guidance, and encouragement I received from my supervisor, colleagues, family, and friends have been invaluable.

Foremost, I extend my deepest appreciation to my supervisor, Dr. Maria Amer. Despite our creative differences, she provided the freedom to pursue research aligned with my interests. I am thankful for her supervision style, which, though challenging, pushed me to strive for excellence in my research and writing. I am especially grateful for her pivotal role in dissuading me from dropping out of the Ph.D. program, a decision I would have regretted. Additionally, I appreciate the extra bursary granted beyond what was initially mentioned in the admission letter. I extend special thanks to Dr. Robert Laganière from the University of Ottawa for graciously accepting our request to serve as the External Examiner. I also extend my thanks to the doctoral examination committee members, Dr. Luis Rodrigues, Dr. Hassan Rivaz, and Dr. Thomas Fevens, for their constructive feedback at various stages of my study program. Furthermore, I am indebted to Dr. Charalambos Poullis for serving as the Arms-length Examiner and to Dr. Javad Dargahi for chairing my defense.

My gratitude extends to the current and former members of the VidPro group, including Mark, Milad, Prateek, Mahdi, Julien, Amin, and Saeid. Interactions with each of you provided me with a fresh perspective on life and helped me recognize and appreciate my privileges. I am thankful to Hassan for his support in conducting simulations related to my research. Special thanks to all members of the High-Performance Computing (HPC) Facility *Speed*, namely Dr. Serguei Mokhov, Dr. Tariq Daradkeh, Gillian Roper, and Dr. Scott Bunnell (former member). Their assistance ensured uninterrupted simulations, especially during tight conference deadlines.

I extend my sincere gratitude to the leadership group at Neoastra Technologies for providing me with a valuable opportunity to work with them during my sabbatical from Ph.D. This experience proved to be enlightening, allowing me to bridge the gap between academic research and customer-centric product development. It significantly influenced my research

trajectory, steering it towards a utility-centric approach. A special acknowledgment goes to the CEO, Ganesh, whose wealth of knowledge and humility had a profound and lasting impact on me. I express my gratitude for the insights gained and hope that life provides another opportunity to collaborate in the future. I would also like to express my heartfelt thanks to my long-time mentor and friend, Prashanth. Your help and support, especially during my early days in Canada, was invaluable. The warmth and hospitality extended by you and your family made my life in Montreal exceptionally joyful.

I extend my heartfelt thanks to my family, whose encouragement, understanding, and patience have been my constant companions throughout my Ph.D. journey. I express deep gratitude to my parents, Gopal and Jyothi, and my sister, Geetha, for their boundless love and unwavering support. Their encouragement has served as a sturdy pillar of strength for me. Last but certainly not least, special thanks go to my wife, Deeksha, for providing emotional support during the highs and lows of my academic pursuit. I cannot emphasize enough how fortunate I am, to have such a caring life partner who comprehends my priorities. I am truly grateful for the sacrifices she has made, both professionally and personally.

In conclusion, I extend my acknowledgment to the numerous individuals who may not be explicitly mentioned but have played pivotal roles in shaping my academic and personal growth. A special note of gratitude goes to *Einzelgänger* for his book "Stoicism for Inner Peace" and the insightful videos on his Youtube channel. His work has played a crucial role assisting me in managing the workload, emphasizing the importance of focusing on the controllables, and embracing the philosophy of *Amor Fati* – to unconditionally accept the outcomes of my actions.

Contents

List of Figures

List of Tables

List of Abbreviations

VOT	Visual Object Tracking
SOT	Single Object Tracking
MOT	Multi-Object Tracking
DCF	Discriminative Correlation Filter
SN	Siamese Network
AOR	Average Overlap Ratio
AUC	Area Under the Curve
FR	Failure Rate
SR	Success Rate
GPU	Graphics Processing Unit
CPU	Central Processing Unit
IoU	Intersection-over-Union
HOG	Histogram of Oriented Gradients
CN	Color Names
CNN	Convolutional Neural Network
SOTA	State-of-the-art
ReLU	Rectified Linear Unit
BN	Batch Normalization

FCN	Fully-Convolutional Network
ViT	Vision Transformer
rtf	Reliable Temporally-consistent Feature adaptation
dca	Dynamic Channel Adaptation
dcp	Dynamic Channel Pruning
MVT	Mobile Vision Transformer-based Tracker
SMAT	Separable Self and Mixed Attention-based Tracker
SMA-ViT	Separable Self and Mixed Attention-based Vision Transformer
ONNX	Open Neural Network Exchange
fps	Frames-per-second
ARC	Aspect Ratio Change
BC	Background Clutter
CM	Camera Motion
DEF	Deformation
FM	Fast Motion
FOC	Full Occlusion
IV	Illumination Variation
LR	Low Resolution
MB	Motion Blur
OV	Out-of-View
POC	Partial Occlusion
ROT	Rotation
SV	Scale Variation
VC	Viewpoint Change

Chapter 1

Introduction

1.1 Visual Object Tracking

Visual object tracking (VOT) is one of the well-studied topics in computer vision, with potential applications in video surveillance, human-computer interaction, augmented reality, robotics, autonomous vehicular navigation, sports analysis, etc. Given a sequence of images, i.e., a video, and a target object tagged in the very first frame, the goal of the visual object tracking algorithm is estimating the trajectory of the target object of any class throughout the sequence. VOT is also known as single object tracking (SOT) since the number of target objects is fixed to one. At first glance, SOT seems like a special case of a multi-object tracking (MOT) algorithm; however, most MOT algorithms are designed to track target objects belonging to a limited set of classes, *e.g.*, pedestrians, and cars. These MOT algorithms [19–21] use off-the-shelf object detectors [22, 23] for target localization and are not suitable for tracking arbitrary objects in videos. In contrast, VOT algorithms are *class-agnostic* since they do not assume any prior knowledge about the class or category of the target object in the video, as depicted in Figure 1.1. Therefore, these algorithms are required to learn the appearance model of the target object *on-the-fly* using the annotation in the first frame and accurately localize the target object in the subsequent frames. Advances in end-to-end trainable learning algorithms [14, 24] and the availability of large-scale training datasets [8–10] have enabled the VOT community to build robust algorithms that can accurately localize arbitrary target objects under the challenges of occlusion, motion blur, deformation, and illumination variations.

The landscape of VOT algorithms is primarily divided into two paradigms: Discriminative Correlation Filters (DCFs) and deep Siamese Networks (SNs) [25]. DCF trackers leverage a combination of hand-crafted and Convolutional Neural Network-based (CNN) features, whereas SN trackers are end-to-end trainable deep models that may utilize CNN or

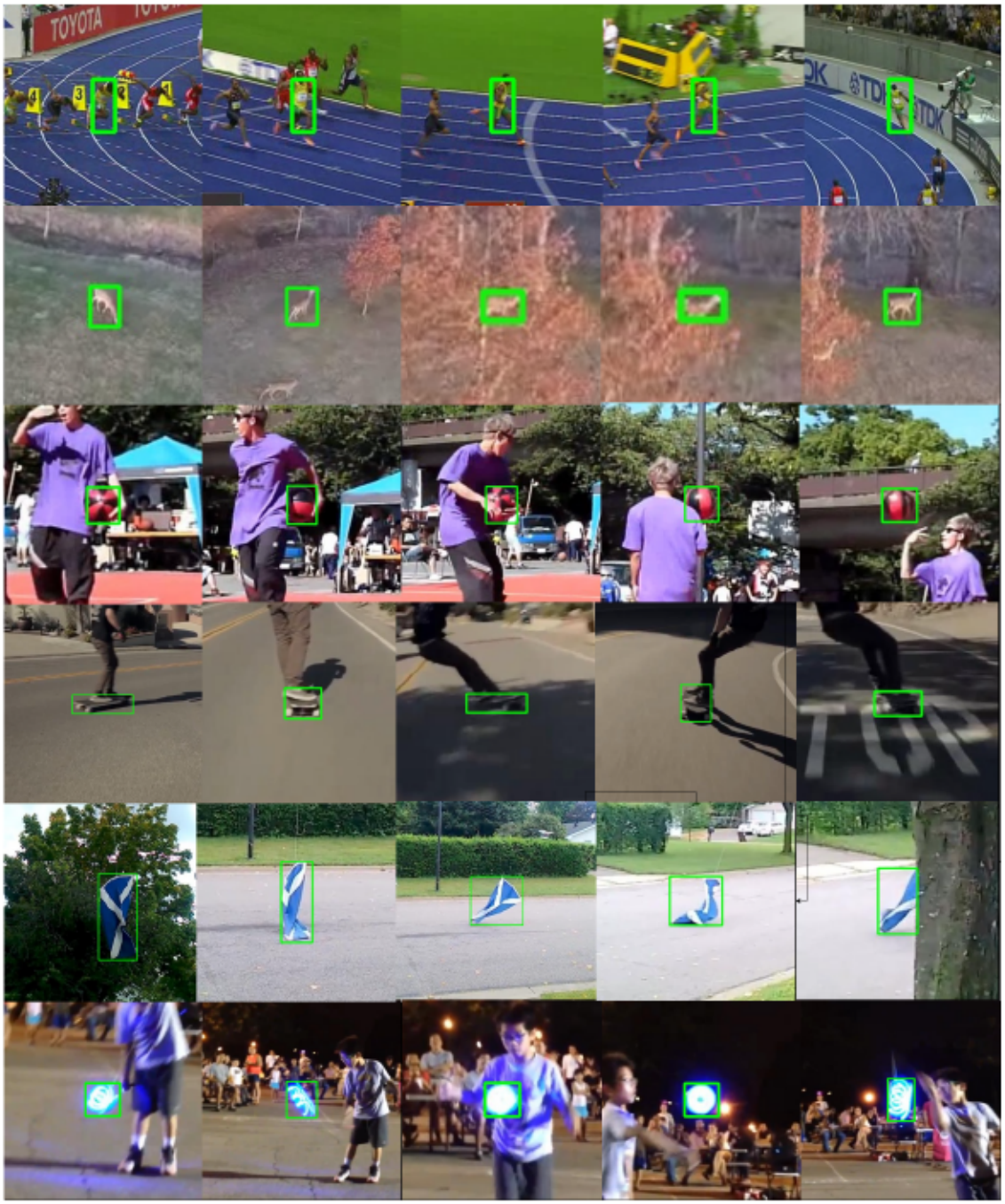

Figure 1.1: Sample videos from VOT datasets showcasing bounding-box annotations encapsulating the target objects. While the first row depicts tracking instances of common object classes such as *human/pedestrian*, rows 2-6 show sequences featuring arbitrary objects such as *deer, basketball, skateboard, flag,* and *yo-yo.* These videos also illustrate challenging scenarios where the target object exhibits fast motion, deformation, and rotation, while contending with external factors like partial occlusion, illumination variation, and background clutter.

transformer blocks (or both) for feature extraction. Both methodologies rely on annotations from the initial frame, termed the target template, to establish the appearance model of the target object. DCF trackers [4, 26, 27] learn target-specific filter coefficients by employing hand-crafted and deep features, whereas SN trackers [13, 28, 29] simply utilize deep features extracted from the annotated template as the target appearance model. In subsequent video frames, both types of trackers search for the target object within a sub-region, also known as the search region, and utilize the learned target model to perform target localization. DCF trackers generate a filter response by correlating features from the search region with learned filter coefficients from the target template to locate the target object. Conversely, SN trackers conduct operations such as naive cross-correlation or deep feature fusion between features extracted from the target template and search region to predict the bounding box coordinates of the target object. While DCF trackers benefit from explicitly learning target-specific filter coefficients, which enhances their robustness against semantically similar background regions, SN trackers excel in speed due to their simpler model architecture, facilitating end-to-end training and testing on a GPU. In recent years, deployment of transformers by SN trackers have showcased state-of-the-art (SOTA) performance across various benchmarks [8–10], leveraging their capacity to learn robust feature representations from extensive annotated data during offline training.

1.2 Approaches to Visual Object Tracking

1.2.1 Discriminative Correlation Filter-based Tracking

DCF is a powerful learning-based tracking approach, whose pipeline is depicted in Figure 1.2. It involves representing the target object, annotated in the first frame, as a set of two-dimensional feature templates (or channels). These channels construct a robust appearance model for the target by learning correlation filter coefficients. The learned model is then employed for target localization in subsequent frames through correlation operations between filter coefficients and features extracted from a defined search region. The target center is determined based on the location of the global maximum in the resulting response map.

To briefly explain the DCF model training and inference, let $\mathbf{X}(t)$ represent the rectangular region enclosing the target in the current frame at time t, and $\hat{\mathbf{X}}(t)$ denotes concatenation of N channels belonging to M different features, i.e., $\hat{\mathbf{X}}(t) = [X_1(t), X_2(t), \ldots, X_N(t)]$. Given $\hat{\mathbf{X}}(t)$, the DCF trackers learn filter coefficients $H_i(t)$, which produce a strong peak at the center of the target object and zero elsewhere, where $H_i(t)$ represent filter coefficients corresponding to channel $X_i(t)$. Next, we omit time index (t) to avoid clutter and use it only

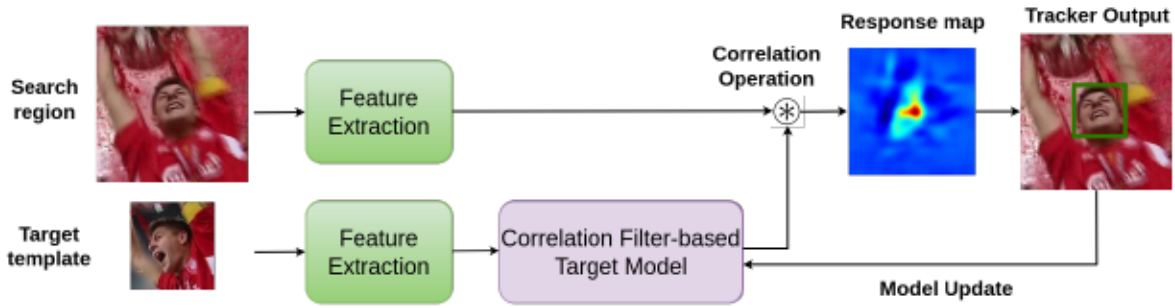

Figure 1.2: Pipeline of DCF-based visual object tracking. The features extracted from the target template are used to learn the correlation filter coefficients. These coefficients are correlated with the features extracted from the search region to perform target localization.

when required. The filter coefficients H_i are learned by solving the ridge regression problem,

$$\underset{H_i}{\text{minimize}} \quad \left\| Y - \left(\sum_{i=1}^{N} X_i \circledast H_i \right) \right\|^2 + \lambda \sum_{i=1}^{N} \|H_i\|^2 , \tag{1}$$

where Y represent the desired DCF response, $\circledast$ denotes the circular correlation operator, and λ the regularization parameter. A Gaussian function with a small variance is typically chosen as Y, which acts as the label function for discriminative model training. By deploying circular correlation operation in Eq. 1, we implicitly use circularly shifted samples of the target template for the discriminative learning of model parameters (i.e., filter coefficients). Furthermore, the circulant nature of training samples facilitates efficiently solving the optimization problem in Eq. 1 in Fourier domain [30] as

$$H_i = \mathcal{F}^{-1} \left(\frac{\overline{\mathcal{F}(Y)} \odot \mathcal{F}(X_i)}{\lambda + \sum_{i=1}^{N} \overline{\mathcal{F}(X_i)} \odot \mathcal{F}(X_i)} \right), \tag{2}$$

where $\mathcal{F}$ denotes the Fourier transform and $\odot$ indicate the pointwise multiplication. To localize the target position at frame $t+1$, a rectangular search region $\mathbf{Z}$ is extracted around the target center at frame t. From the filter coefficients H_i from Eq. 2, the per-channel response C_i is computed as

$$C_i = \mathcal{F}^{-1} \left(\mathcal{F}(H_i) \odot \mathcal{F}(Z_i) \right). \tag{3}$$

Finally, we aggregate these per-channel responses to compute the overall two-dimensional filter response $\mathcal{R}$ as

$$\mathcal{R} = \sum_{i=1}^{N} C_i, \tag{4}$$

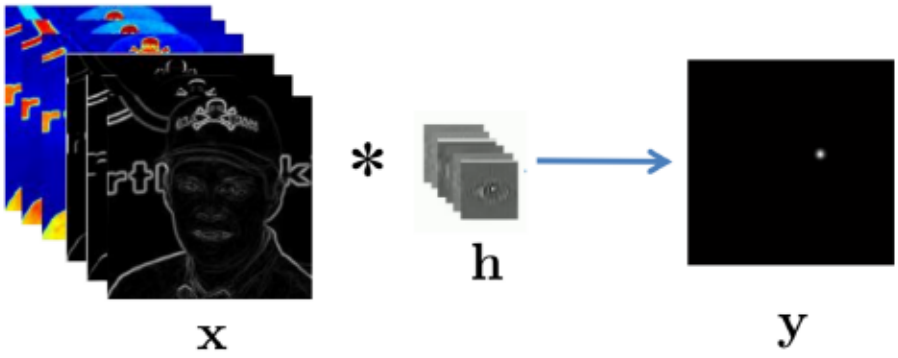

Figure 1.3: Visualization of DCF tracking framework [1]. From the target template, a multi-channel representation X is generated to learn filter coefficients h, which produces an aggregated filter response y with a strong peak at the target center. The same procedure is deployed during tracker inference, where features extracted from the target search region are correlated with the coefficients h to generate the aggregated filter response.

and location of global maximum in $\mathcal{R}$ is determined as target center. The Fourier domain-based computation of filter coefficients in Eq. 1 and per-channel responses in Eq. 3 reduces the overall complexity of learning model parameters to facilitate high tracking speed. Based on the tracker output at t, we update the target appearance model in an auto-regressive manner to compute newly learned filter coefficients.

The DCF framework utilizes diverse channels, ranging from handcrafted features such as Histogram of Oriented Gradients (HOG) [31] and Color Names (CN) [32] to layers of pre-trained Convolutional Neural Networks (CNNs) [33–35], as illustrated in Figure 1.3. These channels capture various low-level (such as shape or color) and high-level visual attributes of the target object. In addition to the idea of deploying powerful feature representations [4,36–41], the DCF framework has seen enhancements over the years due to the advancements in robust target scale estimation [42,43], efficient kernelization [26,44], use of motion features [45, 46], spatial regularization [47, 48], spatio-temporal regularization [16], spatial feature selection [49], joint spatial and channel regularization [50], incorporation of contextual [51] and background [52] information, and adaptive channel weightage learning [53–58].

Despite its flexibility to incorporate multiple channels, the DCF framework could not benefit from end-to-end offline training on large-scale datasets. As a result, these trackers showed suboptimal performance on novel objects unseen by the pre-trained CNN models.

1.2.2 Siamese Network-based Tracking

The SN tracking marks a significant advancement as the first end-to-end trainable framework for VOT. SN trackers conceptualize VOT as a similarity learning problem, wherein they

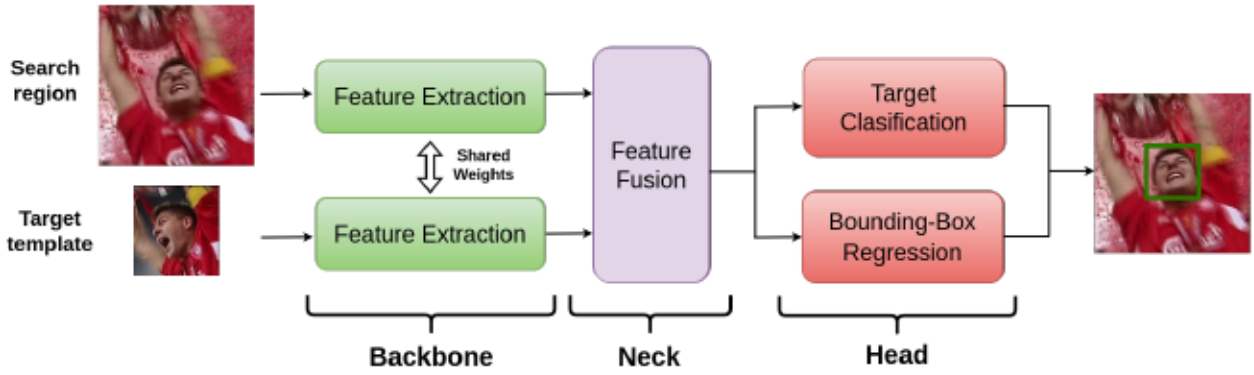

Figure 1.4: Pipeline of Siamese Network-based tracking. The backbone module produces the feature representation of the target object and the search region at a given frame in the video. These features are combined in the neck module to generate a fused feature encoding. The head module decodes the fused encoding to estimate the target object's spatial location and bounding-box coordinates.

learn an embedding space during offline training to maximize the similarity score between template and search regions [24]. By leveraging large-scale external datasets with diverse object classes, SN trackers acquire a generic feature representation that exhibits strong generalization capabilities, especially toward novel object categories. Despite having more model parameters than DCF counterparts, SN trackers manage to run at real-time speed, thanks to the utilization of GPU-friendly modules that harness the parallelization capabilities of the specialized hardware.

The SN tracking pipeline, depicted in Figure 1.4, comprises three key components: a backbone module for feature extraction, a neck module for relational modeling (or feature fusion), and a head module designed for target state estimation. Drawing inspiration from renowned image classification networks such as AlexNet [33], VGGNet [34], and ResNet [35], the backbone module generates a feature representation crucial for accurate and robust target localization in videos. Subsequently, the neck module aggregates features from the template and search regions using cross-correlation or a similar fusion operator, generating a fused feature encoding. Finally, the head module decodes the fused encoding to predict the dense target classification score map, indicating the probability of the target object's presence at different spatial locations in the search region. Optionally, the head may include a regression module for estimating the bounding-box coordinates of the target. Notably, all modules in an SN tracker support end-to-end training, enabling them to learn rich feature representations and complex relationships from extensive training data. In recent years, SN-based trackers have witnessed various advancements, such as employing deeper CNN-based backbones [59–61], advanced feature fusion techniques [62–65], robust target state estimation methods [66–70], hard-negative mining for offline training [71, 72], DCF-like modeling of

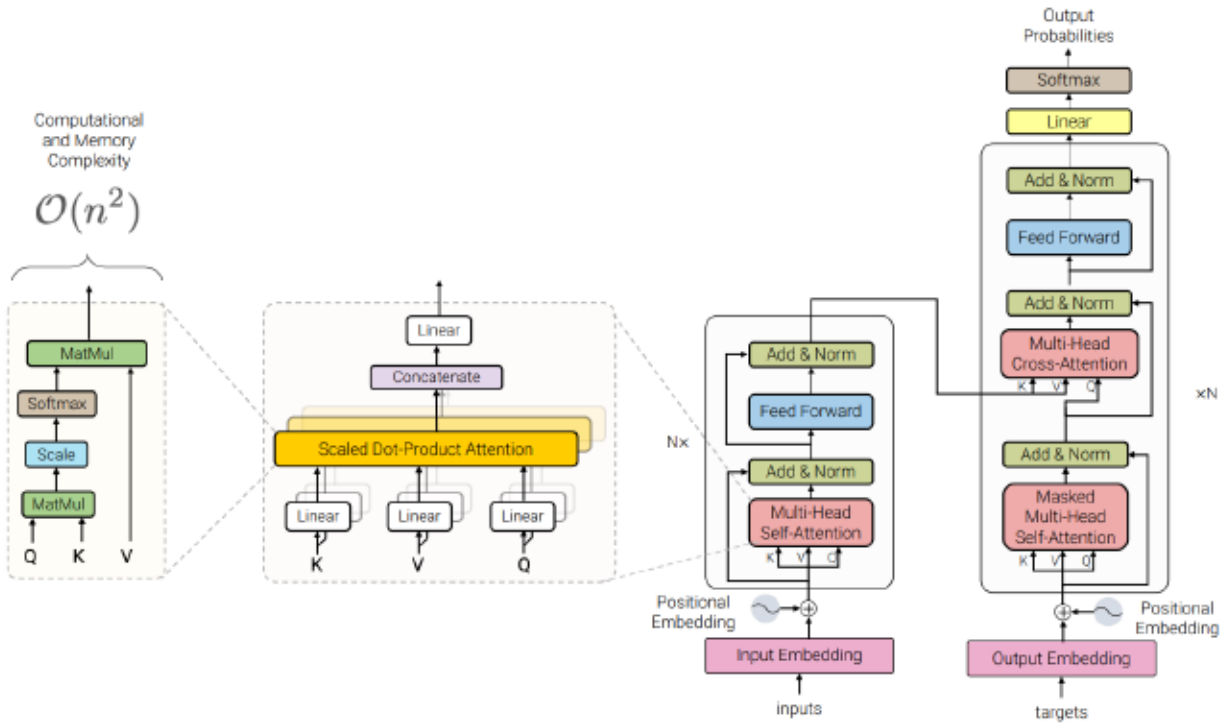

Figure 1.5: Encoder-decoder architecture of the standard transformer module [2] (image from [3]). The multi-head self-attention block learns the global inter-dependencies within the tokenized input data based on the Query $\mathcal{Q}$, Key $\mathcal{K}$, and Value $\mathcal{V}$ generated using the input data.

target object [27,73], knowledge distillation [74], and other architectural innovations [75–83]. These advancements have collectively propelled SN trackers to the forefront, ensuring their continual status as SOTA.

Despite their impressive performance and high speed, deployment of CNN modules restricts an SN tracker's capability to model the global feature interdependencies between the target template and the search region. This limitation degrades the tracker performance on sequences where the target undergoes significant appearance variations and heavy occlusion [13]. To address these limitations, adopting Transformers [2] for SN tracking has emerged as a promising solution, demonstrating superior results compared to traditional CNN-based approaches.

1.2.3 Transformer-based Tracking

Transformer [2] is a differentiable deep learning architecture originally designed for sequence-to-sequence learning tasks in natural language processing (NLP). Illustrated in Figure 1.5, the transformer architecture employs a multi-head self-attention mechanism to learn the

relative importance of each token within a given sequential input data. The self-attention mechanism, represented by Eq. 5, computes the attention scores based on the Query ($\mathcal{Q}$), Key ($\mathcal{K}$), and Value ($\mathcal{V}$) matrices, which are linear projections of the input data X, and d_k indicates the dimensionality of $\mathcal{K}$. The softmax-normalized dot product of $\mathcal{Q}$ and $\mathcal{K}$ is used to weight the values in $\mathcal{V}$. This process captures the interdependencies between different tokens in the sequence.

$$\text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = \text{softmax}\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{d_k}}\right)\mathcal{V} \tag{5}$$

In the context of computer vision, attention mechanism has proven effective in modeling global interdependencies both within (self-attention) and between (cross-attention) input streams. This has led to the application of transformers in various computer vision tasks [84–86], including VOT [14, 15, 87].

The transformer-based trackers initially adopted an encoder-decoder architecture for feature fusion in SN tracking [13,28]. The global contextual modeling provided by self and cross-attention mechanisms improved the quality of fused encodings compared to correlation-based shallow fusion techniques [63], resulting in enhanced tracker performance. Furthermore, the introduction of transformer-based backbones for feature extraction has contributed to achieving SOTA results in VOT benchmarks [8–10, 88–91]. These trackers perform joint feature extraction and fusion within the backbone [14, 15, 92], rendering explicit feature fusion module redundant and simplifying the tracking pipeline.

1.3 Datasets and Performance Metrics

Numerous datasets have been introduced to facilitate the end-to-end training, validation, and testing of VOT algorithms, each serving a unique purpose in assessing the tracker performance under various video challenges. Table 1.1 summarizes key statistics and recommended evaluation metrics for popular visual object tracking datasets. The OTB100 dataset [6] was an early benchmark comprising 100 challenging test sequences with per-sequence attribute annotations such as occlusion and fast motion. The TC128 dataset [93] followed a similar structure with 128 sequences and attribute annotations, providing a diverse set for tracker evaluation. Addressing higher frame rates, the Need-for-Speed (NfS) dataset [90] introduced a collection of 100 sequences captured at 240 frames-per-second (*fps*), alongside a 30 *fps* version (named NfS30) with simulated motion blur. The UAV123 dataset [88] focuses on evaluating tracker performance in aerial videos. The AViST benchmark [89] is a recent addition, offering 120 challenging videos with atmospheric adverse scenarios like rain, fog,

Dataset	# videos	# frames	Dataset split			Attributes	# classes	Evaluation metrics	Server-based evaluation
			Train	Val	Test				
OTB100 [6]	100	**59K**	-	-	100	11	16	*AUC*	No
TC128 [93]	128	**55K**	-	-	128	11	27	*AUC*	No
NfS30 [90]	100	**383K**	-	-	100	9	17	*AUC*, *P*	No
UAV123 [88]	123	**113K**	-	-	123	12	9	*AUC*, *P*	No
AVisT [89]	120	**80K**	-	-	120	18	42	*AUC*, *OP*	No
TrackingNet [10]	30643	**14.4M**	30132	-	511	15	27	*AUC*, *P*, P_{norm}	Yes
LaSOT [9]	1400	**3.87M**	1120	-	280	14	85	*AUC*, *P*, P_{norm}	No
GOT10k [8]	9695	**1.45M**	9335	180	180	6	563	*AO*, *SR*	Yes

Table 1.1: Summary of VOT datasets and their related statistics. It should be noted that the metrics AOR, AO, and AUC are equivalent to one another [12].

fire, low-light, snow, tornado, and smoke, impacting target appearance. The Visual Object Tracking Committee (VOTC) [94] conducts an annual competition with a publicly available dataset of 60 challenging videos, including groundtruth annotations. Additionally, the VOTC hosts a private test dataset with 60 videos and hidden annotations prevents hyper-parameter fine-tuning on the test set. Notably, none of the mentioned datasets include a large-scale training split for end-to-end training of VOT algorithms.

TrackingNet [10], LaSOT [9], and GOT10k [8] stand out as large-scale tracking-specific datasets with train-test splits. TrackingNet, with 30,643 videos, employs a train-to-test split of 30,132-to-511 videos, curated from the YouTube-BoundingBoxes dataset [95]. LaSOT, containing 1,400 videos, adopts a split of 1,120 for training and 280 for testing, along with 14 different attribute annotations. It assesses long-term tracking capabilities with an average video length of 2,500 frames. GOT10k focuses on tracker generalization, presenting a dataset with non-overlapping object categories between the training and test split. Comprising 9,695 videos, GOT10k features training, validation, and test splits of 9,335, 180, and 180, respectively. Both GOT10k and TrackingNet provide an online evaluation server, sequestering test set annotations to ensure fair evaluations.

Several metrics are proposed to assess the performance of VOT algorithms. The most popular metric, Average Overlap Ratio (AOR) or the Average Overlap (AO), quantifies target localization accuracy by measuring Intersection-over-Union (IoU) values between groundtruth and predicted bounding boxes. AOR is equivalent to another popular metric, Area Under the Curve (or AUC), as demonstrated by [12]. The Precision metric (P) measures distance between the groundtruth annotations and tracker output bounding-boxes, in terms of pixels. Since P is sensitive to target size and image resolution, [10] proposed Normalized Precision (P_{norm}), which computes the precision metric on normalized bounding boxes. Success Rate (SR) or Overlap Precision (OP) evaluates the robustness against tracking failures by computing the fraction of frames having an IoU value greater than a threshold τ. The Failure Rate (FR) indicates the fraction of frames with IoU values below

Tracker	Source	Hand-crafted	CNN	total # channels	OTB100 *AUC* ↑	*fps* CPU ↑
LADCF* [49]	TIP'19	✓	✓	553	0.715	1.3
LADCF [49]	TIP'19	✓	✗	41	0.675	18.2
DeepSTRCF [16]	CVPR'18	✓	✓	553	0.683	5.3
STRCF [16]	CVPR'18	✓	✗	41	0.651	24.3
ECO [4]	CVPR'17	✓	✓	93	0.700	12.0
ECOHC [4]	CVPR'17	✓	✗	13	0.650	60.0

Table 1.2: Accuracy versus speed of DCF trackers with hand-crafted and CNN features (or channels). The last two columns indicate the tracker performance on OTB100 dataset and *fps* value measured on a CPU.

the given threshold.

In VOT literature, tracker performance is often categorized in terms of "accuracy" (using metrics such as AOR) and "robustness" (based on metrics such as FR, SR, and P).

1.4 Scopes and Objectives

Two primary challenges in developing an advanced VOT algorithm are accurate localization while tracking arbitrary target objects and robust tracking under occlusion, motion blur, and other video challenges. DCF trackers tackle such challenges by incorporating channel templates extracted from pre-trained CNN models [17, 35] (details are in Section 1.2.1). Integration of these channels enhances the performance of DCF trackers, as illustrated in Table 1.2. However, this improvement comes at the expense of decreased inference speed due to increased computational complexity. For instance, LADCF* demonstrates a 4% higher *AUC* metric on the OTB100 dataset compared to its faster variant LADCF, primarily due to the incorporation of ResNet channels [35]. However, this enhancement results in a 14× decrease in *fps* value on a CPU. In the case of CNN-based DCF trackers in Table 1.2, the substantial increase in the total number of channels, for example, from 41 in LADCF to 553 channels in LADCF*, compromises the computational efficiency of learning filter coefficients in the Fourier domain (*cf.* Eq. 2). This limitation restricts the practical utility of these tracking algorithms for real-time applications, especially those with stringent time constraints such as remote surveillance and augmented reality.

A similar trend is observed in transformer-based SN trackers, where performance gains are achieved at the cost of increased computational complexity. The deployment of transformers as the feature fusion module by [13], and subsequently as the backbone by [14, 15], has led to significant performance improvements compared to CNN-based DiMP-50 [27] and Ocean [68], as highlighted in Table 1.3. Remarkably, alongside the modeling benefits, the

Tracker	Source	GOT10k-test		TrackingNet-test		#params ↓	*fps*	
		$OR \uparrow$	$SR_{0.50} \uparrow$	$AUC \uparrow$	$P_{norm} \uparrow$	(in millions)	GPU ↑	CPU ↑
DiMP-50 [27]	ICCV'19	0.611	0.717	0.740	0.801	26.1	61.5	15.0
Ocean [68]	ECCV'20	0.615	0.735	0.693	0.794	44.3	127.8	10.1
STARK-ST101 [13]	ICCV'21	0.688	0.781	0.820	0.869	47.2	80.0	7.8
OSTrack-384 [14]	ECCV'22	0.740	0.835	0.839	0.885	92.1	74.4	4.4
MixFormer-L [15]	CVPR'22	0.756	0.857	0.839	0.889	183.9	45.2	< 5

Table 1.3: Performance of CNN (DiMP and Ocean) versus transformer-based trackers on GOT10k [8] and TrackingNet [10] test datasets. The deployment of transformers for feature fusion [13] and as the backbone [14, 15] has improved the tracking performance, but at the cost of increased model complexity and latency.

introduction of transformers has resulted in an increase in the number of tracker parameters. Consequently, these trackers struggle to achieve high tracking speeds (not even real time of 30 *fps*) during inference on a CPU and heavily rely on GPUs for real-time performance (i.e., $\geq$ 30 *fps*), as indicated in Table 1.3. This dependency on specialized hardware poses challenges for deploying these algorithms in applications with limited resources (e.g., smart-surveillance, UAV monitoring) that require real-time processing on a general-purpose constrained hardware, such as a CPU.

This thesis tackles the aforementioned challenges and contributes to the existing knowledge by proposing several strategies to enhance the performance of VOT algorithms, encompassing both DCF and SN trackers, while ensuring high tracking speed on general-purpose hardware, such as a CPU.

In the initial stages of this thesis, we focus on improving the performance of DCF-based VOT algorithms through the introduction of adaptive channel weights. The main objective is to mitigate the impact of non-discriminative channels on target localization. To achieve this, we first leverage the concept of channel reliability, which serves as an approximate measure of a channel's discriminative power based on its filter response. Our approach involves transforming these reliability scores into adaptive channel weights, formulated as a convex optimization problem. We then incorporate considerations of inter-channel relations and temporal consistency of learned weights into the optimization process. Our final contribution to DCF tracking involves developing efficient algorithms to solve the optimization problem for channel weights, ensuring minimal disruption to tracker inference speed.

Expanding our research to encompass recent SN trackers, in the later stages of the thesis, we introduce two transformer-based lightweight algorithms designed for efficient object tracking. Within SN tracker models, the backbone assumes particular importance, as it ideally captures both spatially local and long-range dependencies, which is crucial for accurate target localization in the presence of video challenges such as occlusion and target

deformations. To accomplish this goal of generating robust feature representations, the proposed trackers leverage cascaded CNN and transformer blocks in the backbone. This hybrid design combines the strengths of convolutions for learning spatially local representations and transformers for modeling long-range dependencies. In addition to generating effective feature encodings, the significance of the proposed trackers lies in their ability to maintain high inference speed. Given that the backbone module plays a pivotal role in overall latency due to a higher number of model parameters and floating-point operations, our hybrid design using CNNs and transformers strikes a balance between accuracy and latency. This is particularly noteworthy when compared to SOTA fully transformer-based trackers, as our methods exhibit better tradeoff in performance and speed on resource-constrained hardware such as a CPU.

In addition to these technical contributions, we have undertaken a comprehensive evaluation of the proposed tracking algorithms across multiple challenging benchmarks. This evaluation aims to highlight their effectiveness in terms of both speed and performance compared to relevant trackers. Furthermore, we conduct a comparative analysis with SOTA tracking algorithms, encompassing both DCF and SN approaches, elucidating the trade-off between accuracy and speed offered by our efficient tracking methods. To provide a nuanced understanding, we quantify the strengths and limitations of our tracker across different video attributes, including occlusion and fast motion. This is achieved through attribute-based evaluation metrics computed on large-scale VOT datasets. We also perform ablation studies to gain insights into the overall impact of the proposed modules on tracker performance. This analytical approach allows us to systematically assess the contribution of each module to the overall effectiveness of the tracker. Finally, we delve into the benefits of employing suitable inference frameworks or backends to further enhance the *fps* values of our SN trackers on different inference platforms. Leveraging frameworks such as ONNX-Runtime [18] and TensorRT [96], we tap into the computational capabilities of the chosen platform for model deployment. This strategic choice aims to boost inference speed compared to the native framework used for model development and training. Through this analysis, we provide insights into optimizing the deployment of our tracking algorithms on various platforms, enhancing their real-world applicability and performance.

1.5 Thesis Statement

In contemporary deep learning-based trackers, superior tracking performance is often achieved by an escalation in the number of model parameters. While this contributes to enhanced

capabilities, it concurrently introduces higher computational complexity, which, in turn, adversely affects the inference speed of these trackers, limiting their applicability in scenarios with hardware constraints such as a CPU or edge devices. This thesis introduces innovative object-tracking methodologies to achieve high inference speed on resource-constrained hardware while enhancing performance. In DCF tracking, we propose reliability-based adaptive weighing solutions to mitigate the impact of non-discriminative channels on tracker output, along with efficient solutions ensuring faster learning of adaptive weights without compromising inference speed. For SN-based lightweight tracking, we harness the strengths of combining CNNs and transformers in the backbone and head module, for the first time in concurrency, to achieve superior performance at high speed. The thesis also presents a method to further reduce tracker latency by employing computationally efficient separable self and mixed attention transformers as a drop-in replacement for standard transformers.

1.6 Organization of the Thesis

The main chapters 2 to 6 of this thesis are based on our published manuscripts. We also include additional details such as derivations, plots, and connecting texts concerning the coherence and originality of the research reported in the thesis.

- Chapter 2 is from our paper [56], which proposes an convex optimization-based method for DCF tracking with online adaptation of feature weights based on their reliability scores.

- Chapter 3 is from our paper [57]. It extends the aforementioned idea towards an dynamic channel pruning method for DCF tracking.

- Building upon the insights from Chapter 3, Chapter 4 discusses an online channel weight adaptation method, incorporating inter-channel relations to prevent false suppression of discriminative channels, as published in our manuscript [58].

- Chapter 5 is from our paper [97]. It investigates the utility of Mobile Vision Transformers as the backbone for SN-based VOT, presenting a lightweight yet high-performance tracking algorithm named *MVT*.

- Chapter 6 builds upon the advancements made in Chapter 5, further improving the computational efficiency of *MVT* by replacing the standard transformers in *MVT*'s backbone by a separable mixed attention transformer block. We term the resulting method *SMAT* and it is from manuscript [98].

- We summarize the thesis in Chapter 7, offering insights into the findings and potential directions for future research.

- Appendices A and B present the proofs for solving the optimization problems described in Chapter 2 and 3, respectively.

1.7 Publications

This thesis has led to the following publications.

- *Goutam Yelluru Gopal and Maria Amer*, **Separable Self and Mixed Attention Transformers for Efficient Object Tracking** at *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, Waikoloa, Hawaii. (*WACV* is a leading computer vision conference).

- *Goutam Yelluru Gopal and Maria Amer*, **Mobile Vision Transformer-based Visual Object Tracking** at *34th British Machine Vision Conference (BMVC)*, November 2023, Aberdeen, UK. (*BMVC* is a leading computer vision conference).

- *Goutam Yelluru Gopal and Maria Amer*, **Reliable Interconnected Channels for Dynamic DCF based Visual Tracking** *Multimedia Tools and Applications (MMTA)*, Volume 83, pages 839-859, (2024), Springer Journal. Published on May 2023. (*MMTA* is a leading multimedia journal).

- *Goutam Yelluru Gopal and Maria Amer*, **Reliable Temporally Consistent Feature Adaptation for Visual Object Tracking** *IEEE* 27^{th} *International Conference on Image Processing (ICIP)*, October 2020, Abu Dhabi, UAE. (*ICIP* is a leading signal processing conference).

- *Goutam Yelluru Gopal and Maria Amer*, **Dynamic Channel Pruning For Correlation Filter Based Object Tracking** *IEEE* 45^{th} *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, Barcelona, Spain. (*ICASSP* is a leading signal processing conference).

Chapter 2

Reliable Temporally Consistent Feature Adaptation for DCF Tracking

2.1 Introduction

Discriminative Correlation Filter-based (DCF) trackers, designed to operate at real-time speed on a CPU, typically rely on handcrafted features such as Histogram of Oriented Gradients (HOG) [31] and Color Names (CN) [32] [26, 54, 99] to construct the appearance model of the target object. In such trackers, the target is localized based on sum of filter responses corresponding to different features, with fixed per-feature weightage. Some of these features may become non-discriminative or irrelevant over time, due to their susceptibility to video challenges such as target deformation, fast motion, and cluttered background. Consequently, the aggregated DCF response gets corrupted due to noisy responses from non-discriminative features, leading to tracker failure or drift. To mitigate the impact of noisy features on tracker performance, in this chapter, we propose a method for online learning of feature weights, dynamically adapting to the video challenges.

The DCF-based tracker CSRDCF [54] introduced the concept of channel reliability, assessing the discriminative power of a channel based on its filter response. As DCF coefficients aim for a unimodal filter response (*cf.* Figure 1.3), CSRDCF utilized the ratio between the second and first highest non-adjacent peaks in the per-channel response map to gauge channel reliability. Expanding upon this notion, we present a method for online learning of feature weights, grounded in the concept of channel reliability from [54]. Given the heuristic nature of computing reliability scores, which can lead to noisy estimates, we depart from [54] by incorporating the strong correlation between successive frames in a video. This ensures temporal consistency in learned weights, mitigating the impact of noisy reliability estimates.

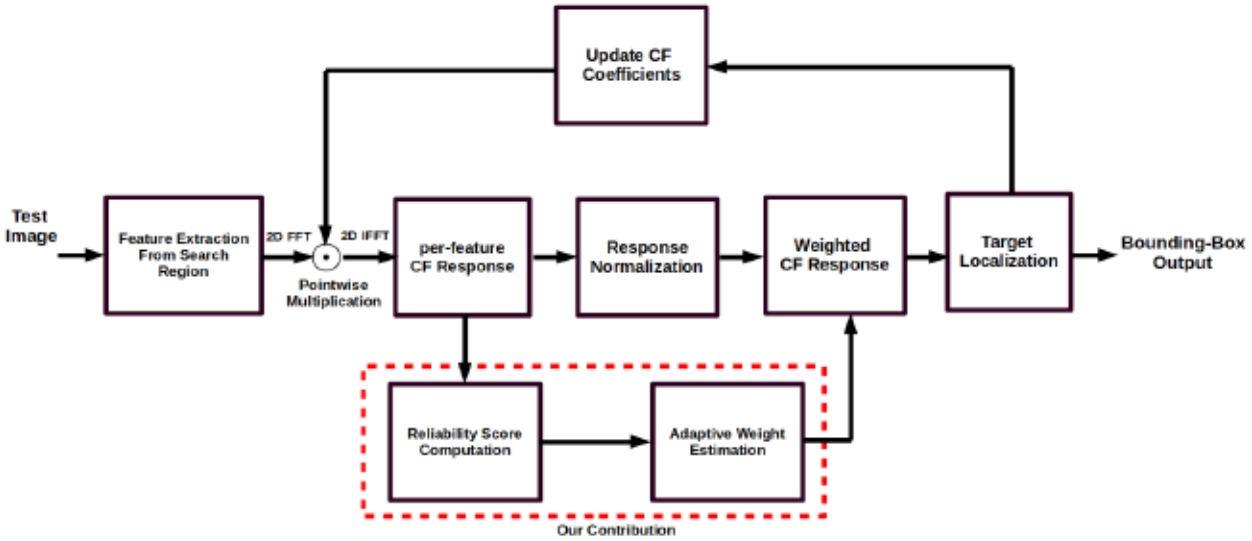

Figure 2.1: Integration of proposed feature adaptation scheme into DCF tracking.

Our contributions encompass:

- An approach for quantifying feature reliability scores derived from per-feature filter responses.
- A method for adaptive learning of feature weights based on reliability scores, maintaining tracking speed. This adaptive feature weight learning is formulated as a convex optimization problem with enforced temporal consistency.
- An algorithm for efficiently solving the optimization problem, leveraging the corresponding Lagrangian dual problem. We propose a solution using cyclic coordinate ascent, providing an expression to compute feature weights through the dual variables.

Additionally, our method can be easily incorporated into any DCF tracker using multiple features (see Figure 2.1). We show the effectiveness of our approach by integrating with two popular DCF trackers: ECOHC [4] and STRCF [16] as baseline.

2.2 Related Work

The DCF-based CSRDCF [54] utilized normalized channel reliability scores as channel weights, while [100] proposed a method for joint learning of CF coefficients and channel weights through least-square estimation. In contrast to both [54] and [100], our approach introduces a robust online learning mechanism for feature weights based on estimated feature reliability scores. However, the bootstrapping process involved in computing reliability scores can lead to noisy estimates, adversely affecting tracking performance. Thus, different

than [54], we model the correlation between successive frames to enforce temporal consistency of learned weights, hence reducing the impact of such noisy estimates. Furthermore, unlike the least-squared estimate in [100], our method estimates reliability scores as a non-linear function of per-feature responses.

2.3 Proposed Method

Figure 2.1 depicts our reliability-based feature weight learning method in the context of DCF tracking. Section 2.3.1 provides a brief description on integrating our method in DCF tracking framework. Section 2.3.2 describes the proposed reliability score estimation from per-feature filter responses. Section 2.3.3 describes our formulation of the optimization problem to learn adaptive weights and propose an algorithm to efficiently solve it (in Section 2.3.4).

2.3.1 Multi-feature DCF Tracking

We modify the mathematical description of DCF tracking in Section 1.2.1 to describe the multi-feature DCF setting. Let X represents the rectangular region enclosing the target object and X_{ij} represents the channel with index j belonging to i^{th} feature. We assume that, there are a total of M features with N_i channels corresponding to i^{th} feature. The problem of learning CF coefficients H_{ij} is defined as

$$\underset{H_{ij}}{\text{minimize}} \quad \left\| Y - \sum_{i=1}^{M} \left(\sum_{j=1}^{N_i} X_{ij} \circledast H_{ij} \right) \right\|^2 + \lambda \left\| H \right\|_F^2 \tag{6}$$

where $||\cdot||_F$ denotes the Frobenius norm, $\circledast$ the circular correlation between X_{ij} and H_{ij}, $\mathcal{F}$ the Fourier transform, and λ the regularization parameter. From the filter coefficients $H_{i,j}$ obtained by solving Eq. 6, the per-feature filter response F_i for the search area Z is computed as

$$F_i = \sum_{j=1}^{N_i} \mathcal{F}^{-1} \left(\mathcal{F}(H_{ij}) \odot \mathcal{F}(Z_{ij}) \right), \tag{7}$$

where Z_{ij} represents j^{th} channel of feature i. In this scenario, the aggregated DCF response $\mathcal{R}$ is computed as the weighted sum of per-feature responses F_i as

$$\mathcal{R} = \sum_{i=1}^{M} \alpha_i F_i, \tag{8}$$

where $\alpha_i \in \mathbb{R}_{\geq 0}$ denote the non-negative weightage corresponding to i^{th} feature. DCF trackers such as ACT [99], ECOHC [4], and STRCF [16] assign equal weightage to all features, i.e., by setting $\alpha_i = 1$ in Eq. 8. Major drawback of using fixed α_i is the effect of unreliable features during video challenges such as background clutter and fast motion, leading to incorrect localization. Therefore, our method focuses on learning α_i at every frame. From feature responses F_i, we estimate their reliability scores (explained in Section 2.3.2) which are used to calculate α_i. Since we use non-linear methods to estimate feature reliability from per-feature responses, joint learning of DCF coefficients H_{ij} in Eq. 7 and feature weights α_i in Eq. 8 leads to non-trivial solution in Fourier domain, which significantly reduces tracking speed. To keep the tracking speed intact, we separately solve for variables H_{ij} and α_i. It also facilitates integration of the proposed method into various multi-feature DCF trackers. Finally, we use learned feature weights α_i to compute the aggregated filter response $\mathcal{R}$ as in Eq. 8.

2.3.2 Feature Reliability Estimation

DCFs are trained to generate a strong, unimodal response at the location of desired target. During video challenges such as rapid target deformation and fast motion, multiple peaks are observed in the aggregated DCF response, due to susceptibility of one or more features to these challenges. Bibi *et al* [101] have demonstrated the sensitivity of DCF tracker to localization error by few pixels, leading to tracking drift. One possible solution is exhaustive search over local maxima of DCF response, which is computationally expensive. An alternative time-efficient approach is to suppress such noisy features based on quantifiable notion of reliability.

In order to estimate feature reliability from per-feature DCF response F_i in Eq. 7, we search for spatial locations of two dominant peaks (known as primary and secondary peak) in F_i. Let (x_p, y_p) be the spatial location of the global maxima (or the primary peak) in two-dimensional feature response F_i. To find the location of secondary peak, we generate a mask S having the same size as i^{th} feature response, centered at the primary peak, as follows

$$S(x, y) = 1 - exp\left\{-\frac{(x - x_p)^2 + (y - y_p)^2}{\sigma^2}\right\}, \tag{9}$$

where (x, y) is the spatial location of a pixel in mask S and parameter σ controls the area of suppression around the primary peak. The mask S is multiplied with per-feature response F_i to suppress the primary peak and highlight the secondary peak. Finally, we estimate

feature reliability score based on relative strengths of primary and secondary peaks as

$$r_i = 1 - \frac{F_i\left(x_s, y_s\right)}{F_i\left(x_p, y_p\right)}, \tag{10}$$

where (x_s, y_s) is the location of secondary peak (i.e., the location of peak value in F_i after applying the mask S). The idea behind Eq. 10 is that, feature responses with unique, stronger peaks are reliable compared to ones having multiple peaks of comparable strength [54].

During our experiments, we observed that the primary peak loses its sharpness and spreads across a larger area as tracking progresses. Hence it is problematic to use a single value of σ in Eq. 9 and define a single, fixed neighborhood for suppression around the primary peak. Therefore, we choose five different values of $\sigma = \{0.05, 0.1, 0.15, 0.25, 0.50\}$ and generate masks to suppress $3 \times 3, 5 \times 5, 7 \times 7, 9 \times 9$ and 11×11 neighborhood. After independently applying these five different masks, we obtain five estimates of the feature reliability score. These scores are averaged using the weights $\{0.0625, 0.0625, 0.125, 0.25, 0.5\}$ respectively, to compute the final estimate for r_i in Eq. 10.

2.3.3 Modeling the Optimization Problem for Adaptive Weight Estimation

Let $\mathbf{r}(t) = \{r_1(t), ..., r_M(t)\}$ represent the reliability scores as per Eq. 10 corresponding to M features used for DCF tracking in the current frame at time t. Let $\boldsymbol{\alpha}(t) = \{\alpha_1(t), ..., \alpha_M(t)\}$ be the adaptive feature weights to be estimated and $\boldsymbol{\beta}(t) = \{\beta_1(t), ..., \beta_M(t)\}$ the *prior* for temporal consistency. We drop the time index (t) to avoid clutter and use it only when required. We define the optimization problem (known as the primal problem) as

$$\underset{\alpha_i}{\text{minimize}} \quad -\sum_{i=1}^{M} \alpha_i r_i + \frac{\gamma}{2} \sum_{i=1}^{M} \left(\alpha_i - \beta_i\right)^2 \tag{11a}$$

$$\text{subject to} \quad \alpha_i \geq 0 \qquad \text{for } i = \{1, \ldots, M\}, \tag{11b}$$

$$\sum_{i=1}^{M} \alpha_i = 1. \tag{11c}$$

The first term in Eq. 11a, along with the constraint in Eq. 11c, assigns higher weights to features with high reliability scores, whereas the second term penalizes large deviation of learned weights with respect to the *prior* $\boldsymbol{\beta}$. The second term in Eq. 11a models the knowledge of strong correlation between successive frames by defining $\boldsymbol{\beta}$ as a function of

feature weights from previous frames. Therefore, it acts as a regularizer by alleviating the impact of noisy reliability scores r_i on the learned weights. γ is the hyperparameter to control the tradeoff between the two objectives. Constraints in Eq. 11b and Eq. 11c ensure that α_i are non-negative and sum to 1, respectively. Prior weights β_i are computed as weighted sum of previous weight estimates as follows

$$\beta_i(t) = \eta\alpha_i(t-1) + (1-\eta)\,\beta_i(t-1), \tag{12}$$

where η is the learning rate to update the prior.

2.3.4 Solution by Dual Coordinate Ascent

The optimization problem in Eq. 11a has no analytical solution and needs to be solved iteratively. One option is to use off-the-shelf solvers such as CVX [102], but we found that it significantly reduces tracking speed. Therefore, we present an algorithm which exploits the structure of the problem and solve it efficiently. To begin with, we derive the Lagrangian dual of Eq. 11a with dual variables $\mathbf{a} = \{a_1, ..., a_M\} \in R^M$, and $b \in R$. The dual problem can be defined as

$$\begin{aligned} \underset{\mathbf{a},\,b}{\text{maximize}} \quad & -\frac{1}{2\gamma}\sum_{i=1}^{M}(-r_i - a_i + b)^2 + \sum_{i=1}^{M}\beta_i(-r_i - a_i + b) - b \\ \text{subject to} \quad & a_i \geq 0, \qquad \text{for } i = \{1, \ldots, M\}. \end{aligned} \tag{13}$$

For the dual problem in Eq. 13, we can see that the objective is quadratic and constraints are separable. It facilitates towards deriving a closed-form solution for each coordinate, hence avoiding the use of gradient-based methods that are generally slower. Derivation of the dual problem can be found in Appendix A. We solve the dual problem in Eq. 13 using the algorithm described in Algorithm 1, which is based on cyclic coordinate ascent [103].

Additionally, we use the condition of strong duality [104] to derive a non-heuristic stopping criterion for our proposed Algorithm 1. Let $\boldsymbol{\alpha}$ and $(\mathbf{a}, b)$ denote arbitrary primal and dual feasible points, respectively. Let $p(\boldsymbol{\alpha})$ and $d(\mathbf{a}, b)$ be the values of primal and dual objectives, obtained by evaluating Eq. 11a and Eq. 13, respectively. If $\boldsymbol{\alpha}^*$ and $(\mathbf{a}^*, b^*)$ are the primal and dual optimal points, the following conditions holds true based on the condition of strong duality,

$$p(\boldsymbol{\alpha}) \geq d(\mathbf{a}, b) \quad \text{and} \quad p(\boldsymbol{\alpha}^*) = d(\mathbf{a}^*, b^*). \tag{14}$$

Hence, we use the rule $p(\boldsymbol{\alpha}) - d(\mathbf{a}, b) \leq \epsilon$ as a stopping criterion to obtain ϵ-suboptimal solution [104]. Upon reaching the stopping criterion, feature weights $\boldsymbol{\alpha}$ are computed using

Input: feature reliability scores: $\mathbf{r}$ and prior weights: $\boldsymbol{\beta}$
Output: feature weights: $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_M\}$
Data:
1 randomly initialize $\mathbf{a}^{(0)} \geq 0$, $b^{(0)}$
2 parameter for stopping criterion: $\epsilon = 10^{-3}$
3 maximum number of iterations: $NumIter = 10^3$
4 iteration index: $k = 0$
5 vector with all entries as 1: $\mathbb{1} \in \mathbb{R}^M$
6 $\langle \cdot, \cdot \rangle$ denotes inner product
7 **while** $k \leq NumIter$ **do**
8 $\quad \mathbf{a}^{(k)} \leftarrow \max\left\{0, -\mathbf{r} + \mathbb{1}b^{(k-1)} - \gamma\boldsymbol{\beta}\right\}$
9 $\quad b^{(k)} \leftarrow \left(\frac{1}{M}\right)\left\langle \mathbb{1}, (\mathbf{r} + \mathbf{a}^{(k)})\right\rangle$
10 $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\beta} - \left(\frac{1}{\gamma}\right)\left(-\mathbf{r} - \mathbf{a}^{(k)} + \mathbb{1}b^{(k)}\right)$
11 $\quad$ Evaluate $p^{(k)}$ and $d^{(k)}$ from Eq. 11a and Eq. 13, respectively.
12 $\quad$ **if** $p^{(k)} - d^{(k)} \leq \epsilon$ **then**
13 $\quad\quad$ break
14 $\quad$ **end**
15 **end**

Algorithm 1: Learning adaptive weights by coordinate ascent

dual variables as shown in step-10 of Algorithm 1.

2.4 Simulation Results

2.4.1 Experimental Setup

We demonstrate the effectiveness of the proposed method on following datasets: short-term VOT2018 [5] (60 test sequences), TC128 [93] (128 test sequences), and NfS30 [90] (100 test sequences). We test our reliability-based, temporally-consistent feature adaptation (*rtf*) method by integrating it into the DCF trackers: ECOHC [4] and STRCF [16]. These baseline trackers are run with hyperparameters provided by their authors, except we do not use dimensionality reduction option for ECOHC. The reason is that ECOHC with dimensionality reduction compresses channels to increase tracking speed. This leads to noisy feature responses, hence reduced tracking accuracy. The parameters of the proposed method are γ of Eq. 11a and η of Eq. 12. We set $\{\gamma, \eta\}$ to $\{5, 0.05\}$ for ECOHC and $\{1.25, 0.01\}$ for STRCF during the experiments, and use the OTB100 dataset to finetune these values. Feature weights at the first frame were initialized to be uniform, i.e., $\alpha_i(0) = 1/M$. All

Trackers	VOT2018		NfS30		TC128		*fps*
	$AOR\uparrow$	$FR\downarrow$	$AOR\uparrow$	$FR\downarrow$	$AOR\uparrow$	$FR\downarrow$	
rtf-ECOHC	**0.3656**	**0.2884**	**0.4209**	**0.2654**	**0.5699**	**0.1386**	31.9
ECOHC	0.3357	0.3480	0.4058	0.3201	0.5556	0.1675	34.8
rtf-STRCF	**0.3588**	**0.3337**	**0.4445**	**0.2590**	**0.5660**	**0.1601**	28.7
STRCF	0.3384	0.3644	0.4241	0.2988	0.5547	0.1683	31.5
CSRDCF	0.3454	0.2958	0.3819	0.3310	0.4888	0.1948	12.0

Table 2.1: Summarizing the feature adaptation *rtf* results for VOT2018, NfS30 and TC128 datasets. *rtf* results are better than base trackers and CSRDCF, and highlighted in **bold**.

Trackers	VOT2018		NfS30		TC128		*fps*
	$AOR\uparrow$	$FR\downarrow$	$AOR\uparrow$	$FR\downarrow$	$AOR\uparrow$	$FR\downarrow$	
rtf-ECOHC	0.3656	0.2884	0.4209	0.2654	0.5699	0.1386	31.9
rtf-STRCF	0.3588	0.3337	0.4445	0.2590	0.5660	0.1601	28.7
ECO	0.3712	0.2819	0.4637	0.2684	0.5855	0.1260	2.5
DeepSTRCF	0.3997	0.2476	0.4672	0.2811	0.5954	0.1475	2.0

Table 2.2: Comparing the proposed *rtf* tracker results with CNN-based DCF trackers.

experiments were run on Intel Xeon CPU E3 3.6GHz, 16GB RAM using MATLAB.

2.4.2 Comparison to Baseline Trackers

We use the evaluation metrics Average Overlap Ratio (AOR) and Failure Rate (FR) to quantify the performance improvement. Due to the deterministic nature of base trackers, we report the results of One Pass Evaluation (OPE). From Table 2.1, we conclude that the proposed *rtf* adaptation improves the performance of the base trackers across all datasets. In case of ECOHC, integration of the proposed *rtf* improves AOR by 2.0% and reduces FR by 4.8% on average across three datasets. Similarly *rtf* improves the performance of baseline STRCF by 1.7% and 2.6% in AOR and FR, respectively, when averaged over the datasets. We can also see that our *rtf*-ECOHC and *rtf*-STRCF outperform the related channel-adaptive CSRDCF, while being faster. Table 2.1 also shows that speed (in terms of *fps*) of the base tracker is minimally affected by the proposed adaptation. It is because the expensive part of generating feature responses is done by the base DCF trackers anyway, and we re-use them to compute reliability scores. Also, the proposed Algorithm 1 efficiently calculates the feature weights without significantly impacting the overall latency.

2.4.3 Comparison to DCF trackers with CNN features

The results presented in Table 2.2 demonstrate that our proposed *rtf* method achieves comparable *FR* values to those of DCF trackers using CNN features, ECO [4] and Deep-STRCF [16]. However, our method operates nearly 15 times faster, providing a superior balance between accuracy and speed.

2.4.4 Subjective Results

Qualitative results in Figure 2.2 showcase the scenarios where our proposed method overcomes tracker failures caused due to various video challenges. In the first two sequences, i.e., *MotorRolling* and *Gymnastics*, the base tracker ECOHC drifts towards the background under the influence of motion blue and rapid deformation, respectively. Similarly, for the *busstation1* and *basketball_1* sequences from row-3 and 4, the base tracker STRCF fails to track the target object due to background clutter. In these cases, the integration of the proposed *rtf* method suppresses the impact of non-discriminate features on the aggregated filter response and avoids tracking failure.

2.5 Summary

DCF tracking with fixed feature weights is prone to failures during video challenges such as rapid target deformation and fast motion. The proposed method overcame this problem by learning adaptive feature weights for DCF trackers at each frame, while enforcing the temporal consistency of learned weights across frames. Our algorithm for solving adaptive feature weights exploited the structure of formulated learning problem and efficiently finds the desired solution. Also, the generic nature of proposed method enabled seamless integration of the our approach into any DCF tracker with multiple features. Due to shared computation of filter responses, our method has limited impact on tracking speed while improving the performance of base trackers. Experimental results using three well-known datasets on CNN-based DCF trackers testify the effectiveness of proposed method towards offering accuracy-speed tradeoff.

Discussing the limitations of the proposed *rtf* method, we acknowledge that our feature adaptation approach relies on at least one reliable feature to track objects effectively. However, when all feature responses are excessively noisy, such as during occlusion, our estimated reliability scores may plummet to very low values (e.g., below 0.3), hence the resulting feature weights are ineffective for accurate tracking. Moreover, each feature employed in DCF tracking comprises a collection of channels. Consequently, assigning a specific weight to a

Figure 2.2: Sample results for ECOHC (row 1-2) and STRCF (row 3-4), showing results of base tracker and proposed *rtf* method.

feature entails providing equal weightage to all channels associated with that feature. This setup presents a disadvantage in scenarios where some channels within a feature are useful to tracking while others are not. One potential solution to this problem involves learning weights for each channel associated with the DCF tracker (as we propose in Chapter 3, based on our paper [57]).

Chapter 3

Dynamic Channel Pruning for DCF-based Object Tracking

3.1 Introduction

The motivation for the proposed dynamic channel weight adaptation and pruning method is grounded in addressing the limitations of the feature adaptation method, *rtf*, discussed in Chapter 2. *rtf* implicitly assigns the same weightage to all channels within a particular feature block, lacking fine-grained control over individual channels. This limitation becomes apparent when tracking failures occur due to noisy responses from multiple channels, especially during video challenges such as rapid target deformation and fast motion.

To quantify this behavior, we analyzed the per-channel localization results of the Discriminative Correlation Filter-based (DCF) tracker ECOHC [4] on the OTB100 [6] dataset. The analysis involved computing the overlap ratio (or IoU) between per-channel bounding box output and groundtruth annotations. The distribution of per-channel IoU, depicted in Figure 3.1, revealed that a significant fraction of channels (approximately 33%) exhibited poor target localization results, with IoU less than 0.5. Furthermore, Figure 3.2 provides a visualization of incorrect target localization (or drift) of a multi-channel DCF tracker ECOHC [4] for the *ants1* sequence. The per-channel responses are mostly unimodal at the beginning, but as the video progresses, several channels lose their discriminative power and exhibit multimodal responses, resulting in tracker drift. Such tracking failures call for online adaptation of channel weights to suppress the effect of non-discriminative channels on target localization and enhance robustness against tracking failures. These observations motivate the proposed dynamic channel weight pruning method to address the impact of non-discriminative channels on the tracker's output.

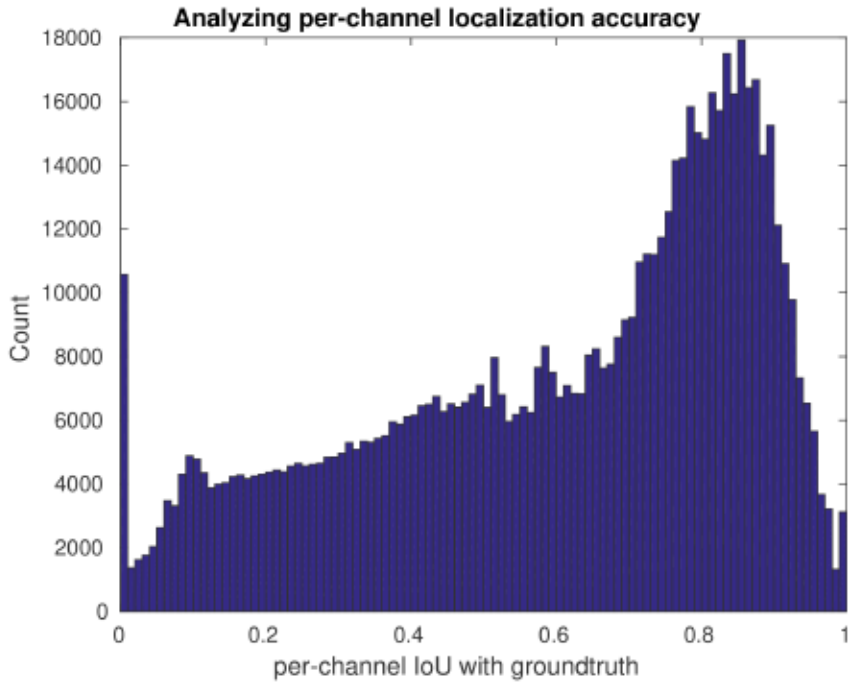

Figure 3.1: Histogram depicting the distribution of per-channel overlap with groundtruth annotations for ECOHC [4] tracker. We observe that a significant fraction of channels fail to locate the target object accurately.

3.2 Related Work

Several existing solutions aimed at mitigating the impact of noisy channels on tracker output. For instance, CART [105] learns channel weights at the first frame and maintains them fixed for the remainder of the sequence, while CPT [106] suggests a method for channel pruning by discarding invalid channels in the initial frame. In contrast to both CART [105] and CPT [106], our approach performs channel adaptation and pruning online, meaning it adjusts channel weights dynamically at every frame to cope with evolving video challenges. Among the methods that adapt the channel weights online, CSRDCF [107] introduced the idea of using reliability scores estimated from per-channel filter responses as channel weights. Different than [107], we model the proposed channel adaptation and pruning method as a convex optimization problem. Additionally, we incorporate the prior knowledge of strong correlation between successive frames in order to reduce the impact of noisy reliability estimates on learned weights. Building on the reliability-based *rtf* method from Chapter 2, the proposed approach in this Chapter 3 introduces the following key components:

- The proposed method includes an online learning approach for channel weight adaptation and dynamic pruning. This process is formulated as a non-smooth convex

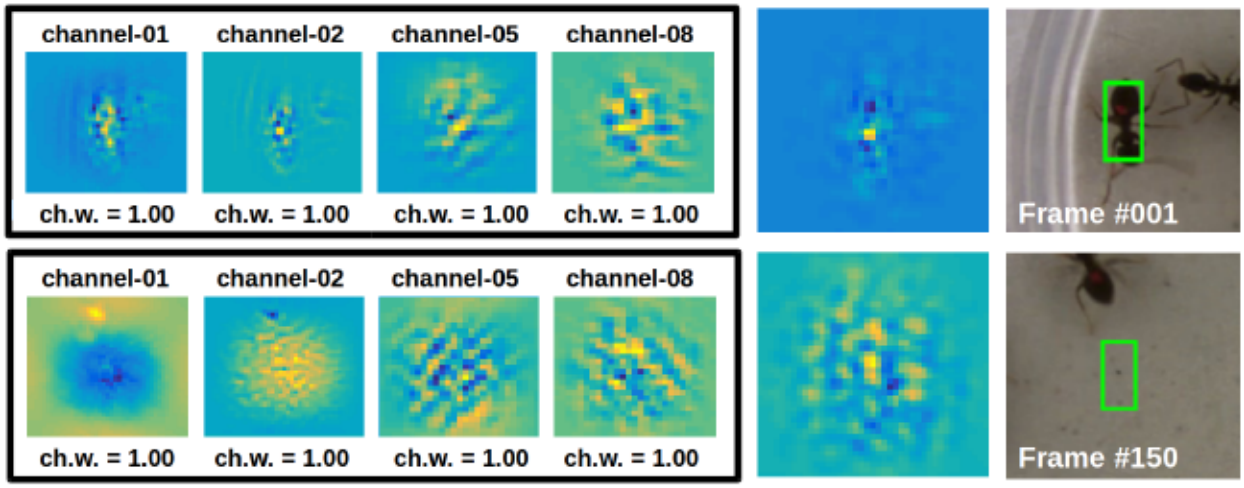

Figure 3.2: Visualization of probable incorrect target localization (or drift) of a multi-channel DCF tracker ECOHC [4] for *ants1* sequence. The first column shows per-channel response samples for ECOHC, with indices and channel weights. The aggregated filter responses, computed as the weighted sum of per-channel responses in each row, are shown in the middle column. The last column shows tracker output based on aggregated filter responses. We observe from the first row that the per-channel responses are mostly unimodal, and their peaks overlap with the target center at frame #1. As the video progresses, at frame #150, several channels lose their discriminative power and exhibit multimodal responses, resulting in tracker drift. This problem motivates the proposed dynamic channel weight pruning to alleviate the impact of non-discriminative channels on tracker output.

optimization problem, while incorporating temporal consistency constraints on channel weights. This allows the method to adapt to changing conditions throughout the tracking sequence.

- To solve the optimization problem for channel weights efficiently, the proposed method leverages the corresponding Lagrangian dual problem. We design an algorithm for the efficient solution of the dual problem, employing block coordinate ascent. This algorithm provides an effective means of minimizing the non-smooth convex objective function. Additionally, an expression is derived to compute channel weights using the dual variables.

3.3 Proposed Dynamic Channel Pruning

From the discussion on multi-channel DCF tracking in Section 1.2.1, let C_i be the filter response corresponding to the i^{th} channel during tracker inference. The aggregated DCF

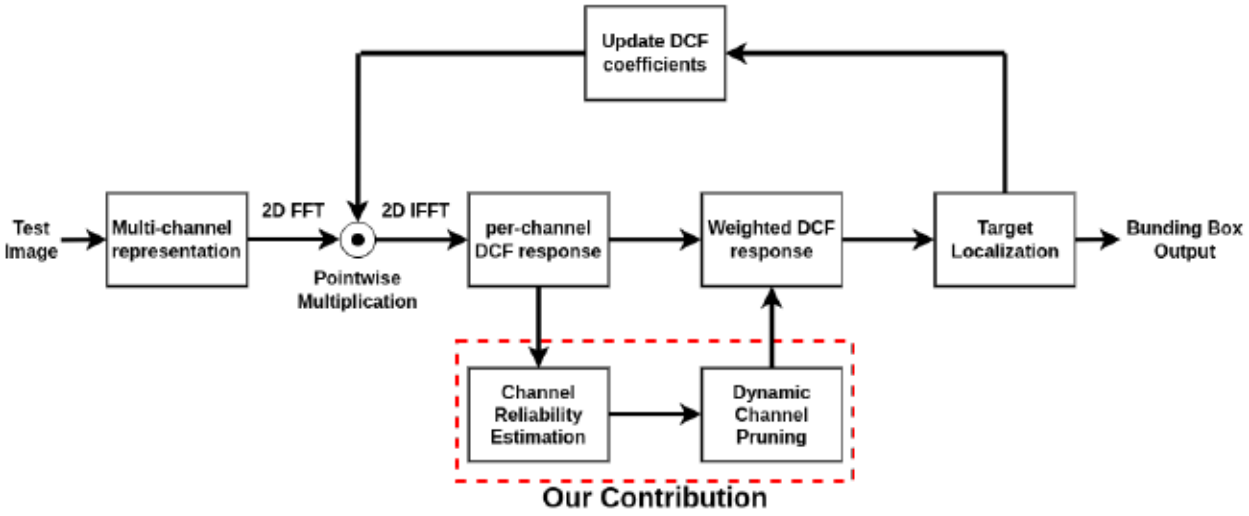

Figure 3.3: Integration of proposed reliability-driven channel pruning method into DCF framework. The overall response is computed as the weighted sum of per-channel responses, where the learned weight of a channel is relative to its reliability score and a temporal prior.

response $\mathcal{R}$ is computed as weighted sum of per-channel responses C_i as

$$\mathcal{R} = \sum_{i=1}^{N} \alpha_i C_i, \tag{15}$$

where $\alpha_i \in \mathbb{R}_{\geq 0}$ denote the non-negative weightage corresponding to the i^{th} channel. The limitation of using fixed α_i values is the potential impact of unreliable channels on the aggregated filter response, as illustrated in Figure 3.2. The proposed method addresses this limitation by learning α_i at every frame, allowing the pruning of unreliable channels by setting the corresponding channel weights α_i to zero. The flexibility of the proposed method enables its integration into various multi-channel DCF trackers, as depicted in Figure 3.3. Using the learned channel weights α_i, the final DCF response $\mathcal{R}$ is computed using the formula in Eq. 15.

3.3.1 Modeling Channel Pruning as an Optimization Problem

Let $\mathbf{r}(t) = \{r_1(t), ..., r_N(t)\}$ represents per-channel reliability scores in the current frame at time t. To compute these reliability scores from filter responses, we follow the same approach described in Section 2.3.2. Let $\boldsymbol{\alpha}(t) = \{\alpha_1(t), ..., \alpha_N(t)\}$ the channel weights to be estimated and $\boldsymbol{\beta}(t) = \{\beta_1(t), ..., \beta_N(t)\}$ denotes *prior* weights for temporal consistency. We define the

optimization problem as

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \quad -\frac{1}{N}\langle \boldsymbol{\alpha}, \mathbf{r} \rangle + \frac{\gamma_1}{2} \|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2^2 + \frac{\gamma_2}{N} \|\boldsymbol{\alpha}\|_1 \tag{16a}$$

$$\text{subject to} \quad 0 \leq \boldsymbol{\alpha} \leq \mathbb{1}, \tag{16b}$$

$$p \cdot N \leq \langle \mathbb{1}, \boldsymbol{\alpha} \rangle. \tag{16c}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product and $\mathbb{1} \in R^N$ is a column vector with all entries as 1. We drop the time index (t) for tidiness and use it whenever necessary. First term in Eq. 16a assigns higher weights to channels with high reliability scores, whereas the second term penalizes large change in channel weights between frames. By modeling strong correlation between successive frames, second term provides stability against noisy reliability scores r_i. Last term in Eq. 16a promotes sparsity of weight vector $\boldsymbol{\alpha}$ by penalizing its ℓ_1-norm [108]. This term facilitates dynamic *pruning* of channels with low reliability scores, by setting corresponding channel weights to zero. Parameters γ_1 and γ_2 control the emphasis on temporal consistency and channel weight sparsity terms, respectively. The constraint in Eq. 16b restrict the range of α_i between $[0, 1]$, and constraint in Eq. 16c asserts our method to select at least $p \cdot N$ channels at each frame, where $p \in [0, 1]$ is a parameter. Use of constraint in Eq. 16c partially reduces sensitivity of the proposed method to parameter γ_2 in Eq. 16a, which otherwise causes pruning of too many channels, as observed in our experiments. Channel weights at time $t - 1$ are used as prior weights at t in Eq. 16a, i.e., $\boldsymbol{\beta}(t) = \boldsymbol{\alpha}(t - 1)$.

3.3.2 Solving the optimization problem

It is a challenging task to efficiently solve the optimization problem in Eq. 16a, due to the non-differentiable term with ℓ_1 norm. Use of off-the-shelf solvers such as CVX [102] significantly reduces tracking speed as we observed during our simulations. We thus derive an equivalent problem to Eq. 16a with differentiable objective by introducing an auxiliary variable $\boldsymbol{\rho} \in \mathbb{R}^N$ and additional linear inequalities [108]. Resulting optimization problem

with a differentiable objective (also referred as the primal problem) can be defined as

$$\begin{aligned} \underset{\boldsymbol{\alpha},\,\boldsymbol{\rho}}{\text{minimize}} \quad & -\frac{1}{N}\langle \boldsymbol{\alpha}, \mathbf{r}\rangle + \frac{\gamma_1}{2}\left\|\boldsymbol{\alpha}-\boldsymbol{\beta}\right\|_2^2 + \frac{\gamma_2}{N}\langle \mathbb{1}, \boldsymbol{\rho}\rangle \\ \text{subject to} \quad & 0 \le \boldsymbol{\alpha} \le \mathbb{1}, \\ & p \cdot N \le \langle \mathbb{1}, \boldsymbol{\alpha}\rangle, \\ & -\boldsymbol{\alpha} \le \boldsymbol{\rho}, \\ & \boldsymbol{\alpha} \le \boldsymbol{\rho}. \end{aligned} \tag{17}$$

The optimization problem in Eq. 17 has no analytical solution and needs to be solved iteratively. Therefore, we propose a method which exploits the structure of primal problem in Eq. 17 and solve it efficiently. To begin with, we derive the Lagrangian dual of the convex optimization problem in Eq. 17 with dual variables $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \in \mathbb{R}^N$ and $d \in \mathbb{R}$. The dual problem can be defined as

$$\begin{aligned} \underset{\mathbf{a},\,\mathbf{b},\,\mathbf{c},\,d}{\text{maximize}} \quad & -\frac{1}{2\gamma_1}\left\|-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} + 2\mathbf{c} - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right\|_2^2 \\ & + \left\langle \boldsymbol{\beta}, -\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} + 2\mathbf{c} - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right\rangle - \langle \mathbb{1}, b\rangle + d \cdot p \cdot N \\ \text{subject to} \quad & 0 \le \mathbf{a}, \\ & 0 \le \mathbf{b}, \\ & 0 \le \mathbf{c} \le \frac{\gamma_2}{N}\mathbb{1}, \\ & 0 \le d. \end{aligned} \tag{18}$$

The derivation of the above dual problem is in Appendix B.

In Eq. 18, the objective is quadratic and constraints are separable, which facilitates deriving a closed-form solution for variable blocks $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, d\}$, thereby avoiding use of slower gradient-based methods. We apply block coordinate ascent [103] to maximize the dual objective, as described in Algorithm 2. Also, we derive a stopping criterion [104] based on conditions of strong duality. Let $f(\boldsymbol{\alpha})$ and $g(\mathbf{a}, \mathbf{b}, \mathbf{c}, d)$ be the value of primal and dual objectives obtained by evaluating Eq. 17 and Eq. 18, respectively. Then, the difference $f(\boldsymbol{\alpha}) - g(\mathbf{a}, \mathbf{b}, \mathbf{c}, d)$ indicates duality gap, which is used for designing the stopping criterion to obtain ϵ-subpotimal solution [104] (shown in line 13 of Algorithm 2). Finally, we compute channel weights using the dual variables as per step-11 in Algorithm 2).

Input: channel reliability scores: $\mathbf{r}$ and prior weights: $\boldsymbol{\beta}$

Output: Channel weights $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_N\}$

Data:

$\{\mathbf{a}^{(0)}, \mathbf{b}^{(0)}, d^{(0)}\} \geq 0, \quad \frac{\gamma_2}{N}\mathbb{1} \geq \mathbf{c}^{(0)} \geq 0,$

maximum number of iterations $NumIter = 10^3$

parameter for stopping criterion: $\epsilon = 10^{-3}$

vector with all entries as 1: $\mathbb{1} \in \mathbb{R}^M$

iteration index: $k = 0$

while $k \leq NumIter$ **do**

$$\mathbf{a}^{(k)} \leftarrow \max\left\{0, -\frac{1}{N}\mathbf{r} + \mathbf{b}^{(k-1)} + 2\mathbf{c}^{(k-1)} - \left(\frac{\gamma_2}{N} + d^{(k-1)}\right)\mathbb{1} - \gamma_1\boldsymbol{\beta}\right\}$$

$$\mathbf{b}^{(k)} \leftarrow \max\left\{0, \frac{1}{N}\mathbf{r} + \mathbf{a}^{(k)} - 2\mathbf{c}^{(k-1)} + \left(\frac{\gamma_2}{N} + d^{(k-1)}\right)\mathbb{1} + \gamma_1\left(\boldsymbol{\beta} - \mathbb{1}\right)\right\}$$

$$\mathbf{c}^{(k)} \leftarrow \max\left\{0, \min\left\{\frac{\gamma_2}{N}, \frac{1}{2}\left(\frac{1}{N}\mathbf{r} + \mathbf{a}^{(k)} - \mathbf{b}^{(k)} + \left(\frac{\gamma_2}{N} + d^{(k-1)}\right)\mathbb{1} + \gamma_1\boldsymbol{\beta}\right)\right\}\right\}$$

$$d^{(k)} \leftarrow \max\left\{0, \frac{1}{N}\left(\left\langle \mathbb{1}, \left(-\frac{1}{N}\mathbf{r} - \mathbf{a}^{(k)} + \mathbf{b}^{(k)} + 2\mathbf{c}^{(k)} - \frac{\gamma_2}{N}\mathbb{1}\right)\right\rangle - \gamma_1\left(\langle \mathbb{1}, \boldsymbol{\beta}\rangle - p \cdot N\right)\right)\right\}$$

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\beta} - \frac{1}{\gamma_1}\left(-\frac{1}{N}\mathbf{r} - \mathbf{a}^{(k)} + \mathbf{b}^{(k)} + 2\mathbf{c}^{(k)} - \left(\frac{\gamma_2}{N} + d^{(k)}\right)\mathbb{1}\right)$$

Evaluate $p^{(k)}$ and $d^{(k)}$ from Eq. 17 and Eq. 18, respectively.

if $p^{(k)} - d^{(k)} \leq \epsilon$ **then**

break

end

end

Algorithm 2: Dual block coordinate ascent for channel weight learning

3.4 Simulation Results

3.4.1 Experimental Setup

We test our dynamic channel pruning (*dcp*) method by integrating it into DCF trackers ECOHC [4] and STRCF [16]. Parameters of the proposed method are γ_1, γ_2, and p of Eq. 16a. We set $\{\gamma_1, \gamma_2, p\}$ to $\{0.025, 0.20, 0.60\}$ for ECOHC and $\{0.025, 0.20, 0.50\}$ for STRCF during the experiments. We use the OTB100 dataset [6] to finetune the values of these hyperparameters. Channel weights at the first frame were initialized to be 0.5, i.e., $\alpha_i(0) = 0.5$. Under this setting, at each frame, our method computes 42 and 41 per-channel weights

Trackers	VOT2018		TC128	
	AOR↑	FR↓	AOR↑	FR↓
dcp-ECOHC (ours)	**0.3701**	**0.2873**	0.5657	0.1462
rtf-ECOHC	0.3656	0.2884	**0.5699**	**0.1386**
ECOHC	0.3357	0.3480	0.5556	0.1675
dcp-STRCF (ours)	**0.3622**	**0.3202**	0.5627	**0.1573**
rtf-STRCF	0.3588	0.3337	**0.5660**	0.1601
STRCF	0.3384	0.3644	0.5547	0.1683
CSRDCF	0.3454	0.2958	0.4888	0.1948

Table 3.1: Proposed Dynamic Channel Pruning *dcp* versus related methods. Results of *dcp* are better than baseline trackers and channel-adaptive CSRDCF. Tracker results of our feature adaptation method (*rtf*) from Chapter 2 are also included. Best results are highlighted in **bold**.

for STRCF and ECOHC (without using dimensionality reduction operator), respectively. We use the VOT2018 [5] and TC128 [93] datasets, and use the evaluation metrics, Average Overlap Ratio AOR and Failure Rate FR. We use the originally provided bounding-boxes as groundtruth during evaluation (i.e., polygonal for VOT2018 and rectangular for TC128). Due to the deterministic nature of base trackers, we report the results of One Pass Evaluation (OPE). Our simulations were conducted on a machine having Intel Xeon CPU without any specialized hardware, such as a GPU.

3.4.2 Comparison to Baseline Trackers

Table 3.1 summarizes the performance of the proposed *dcp* method compared to the baseline trackers. From Table 3.1, we conclude that our *dcp* scheme improves the performance of base trackers in AOR and FR metrics. In case of ECOHC, the integration of the proposed method improves the AOR by 1.8% and reduces FR by 4%, when averaged across the two datasets. Similarly, *dcp*-STRCF exhibits superior AOR and FR by 1.2% and 2.8%, respectively, compared to the baseline STRCF. Table 3.1 also shows that, proposed channel pruning performs better than CSRDCF [107] for the two datasets; where *dcp*-ECOHC achieves the best result. With off-the-shelf solvers such as CVX to solve the minimization problem in Eq. 16a, the proposed *dcp* adaptation for the two base trackers runs at a speed of 2 *fps*. With the use of Algorithm 2, our *dcp*-ECOHC and *dcp*-STRCF achieve a speed of 15 and 12 *fps*, respectively; CSRDCF runs at 12 *fps*. Compared to the deep learning based ECO [4], our *dcp*-ECOHC method has comparable accuracy but, ECO runs at only 2 *fps*.

Figure 3.4: Sample results showing bounding box output of the base tracker and that of our *dcp* adaptation

3.4.3 Qualitative Results

The qualitative results in Figure 3.4 demonstrate scenarios where the proposed method exhibits accurate target localization under various video challenges. In the *drone-across* and *ants1* sequence, the target object exhibits fast motion and video has a cluttered background. For the *motocross2* and *iceskater_1* sequences, the target object is affected by motion blur and rapid deformation. The baseline trackers fail to overcome these challenges, leading to tracking failures. With the integration of proposed *dcp* method, these trackers successfully locate the target object without tracking failures.

3.5 Summary

Under the scheme of uniform channel weights for DCF tracking, presence of noisy, unreliable channel responses deteriorates tracking results. The proposed reliability-based method

negates the impact of unreliable channels by dynamically pruning them. The proposed method has the flexibility to be integrated into many multi-channel DCF trackers. Our algorithm to compute the channel weights exploits the structure of formulated learning problem and efficiently finds the desired solution. Experimental results using baseline DCF trackers testify the effectiveness of the proposed method.

A limitation of the proposed *dcp* method is that the learned weights are sensitive to choice of parameter γ_2, which led to over-pruning of channels in some sequences and led to inferior results. To address this limitation, we introduced the hyperparameter p in Eq. 16c, which controls the percentage of pruned channels. However, this is not a sophisticated solution as the value of hyperparameter p tuned on the entire dataset maybe suboptimal for a particular sequence. In the next Chapter 4, we propose solutions to this limitation.

Chapter 4

Reliable Interconnected Channels for Dynamic DCF Tracking

4.1 Introduction

In Chapter 3, we introduced a pruning-based dynamic channel weight adaptation method, referred to as *dcp*. However, *dcp* encountered a limitation related to the sensitivity of learned weights to the hyperparameter γ_2 in Eq. 16a, resulting in excessive pruning of channels and subsequent tracking failures under some challenges such as occlusion and motion blur. To delve deeper into this issue, we conducted an analysis of the per-channel filter responses and corresponding target localization results for the multi-channel Discriminative Correlation Filter-based (DCF) tracker, ECOHC [4]. Our analysis revealed that certain channels, exemplified by *channel-2* in Figure 4.1, exhibited accurate target localization despite being assigned low reliability scores due to multiple local maxima in their filter responses. Consequently, our channel pruning approach from Chapter 3 erroneously suppressed such discriminative channels, leading to tracking failures. Moreover, our investigation highlighted instances where a discriminative channel displayed a weak global maximum, resulting in low reliability scores according to the ratio of non-adjacent primary and secondary peaks in Eq. 10 (which estimates feature reliability scores r_i). This phenomenon also contributed to the suppression of valuable channels.

To overcome the challenge of erroneously suppressing discriminative channels, we introduce a novel objective function that integrates inter-channel relationships during the online learning of channel weights. This new objective, detailed in Eq. 23, aims to enhance the channel weight adaptation process and alleviate the impact of noisy reliability scores r_i on

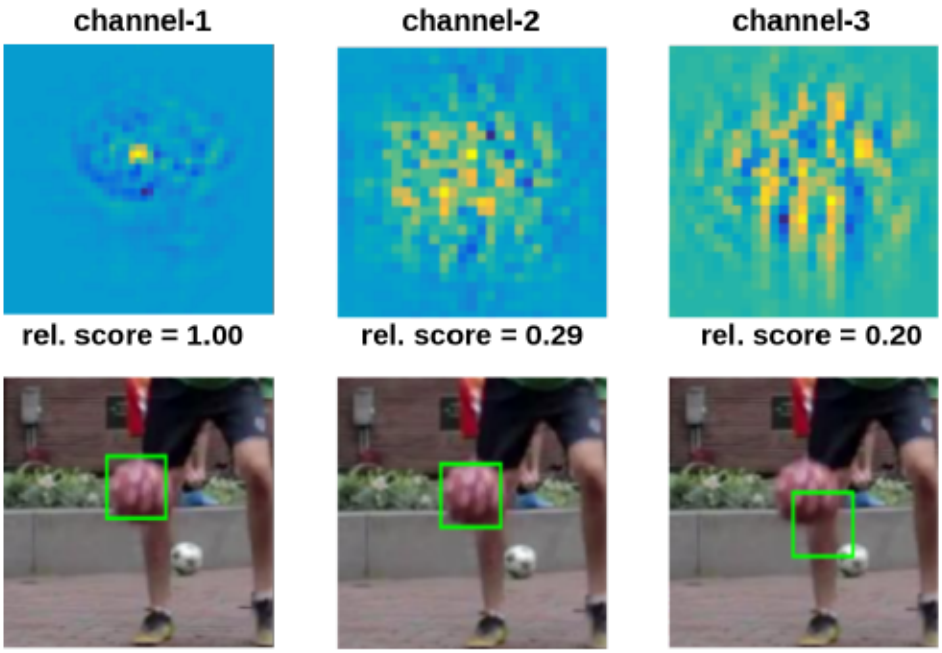

Figure 4.1: Visualization of per-channel filter responses (top row) and corresponding target localization results (bottom row) for the DCF tracker ECOHC [4] on *ball* sequence from VOT2018 dataset [5]. We can see that *channel-2* is assigned low reliability scores due to its multimodal filter response, however it accurately localizes the target object. Hence, our channel pruning method from Chapter 3 falsely suppresses *channel-2* despite it being useful for target localization.

the learned channel weights by considering the interactions and dependencies among channels. By optimizing this objective, our proposed method seeks to offer a more nuanced and context-aware channel weight adaptation approach, ultimately enhancing overall tracking performance and addressing the identified limitations of the initial method presented in Chapter 3. Furthermore, in this Chapter 4, we present an enhanced channel reliability estimation method, building upon our previous approach outlined in Section 2.3.2, to effectively handle scenarios involving channels with weak global maxima.

4.2 Related Work

Related work on channel adaptation can be broadly separated into three categories: rule-based [109–112], norm-based [50, 113–115], and reliability-score based methods [55, 116]. Another way to classify them is into channel-selective (i.e., compute binary channel weights [50, 109–111, 113, 114]) and channel-adaptive (i.e., learn continuous channel weights [55, 112, 115, 116]) trackers. Our proposed method falls under the reliability-score based and channel-adaptive category.

Rule-based method proposed by Che et al. [109] and Zhu et al. [110], aim to select advantageous CNN channels based on feature maps from the first frame of a video. Similarly, [110] proposed a backward channel selection method to drop redundant channels using the first annotated frame. [111] used a channel effectiveness measure based on spatial discrimination and temporal stability, to iteratively prune channels using the first two video frames. [112] proposed an online hedging strategy to assign different weightage to feature responses.

Among norm-based methods, [113] proposed an ℓ_1 regularization method to prune redundant channels. GFSDCF [50] presented a joint learning method for DCF tracking by enforcing group sparsity on spatial and channel dimensions with a temporal regularizer. ACSDCF [114] proposed a group elastic net regularization for joint learning of DCF coefficients and channel selection while maintaining the temporal smoothness of selected channels across frames. CGRCF [115] introduced the idea of modeling inter-channel relationships during channel weight learning by proposing an ℓ_2 norm-based graph regularization objective function to assign matching weights to similar channels.

The reliability-score based CSRDCF [55] introduced the idea of learning channel weights based on reliability scores estimated from DCF responses. It defined channel reliability based on the magnitude of two dominant peaks in per-channel DCF response and computed channel weights by normalizing the estimated reliability scores. JCRCF [116] used peak-to-average correlation energy measure for joint learning of channel weights and CF-coefficients.

In contrast to [109–111, 113], our method periodically evaluates the reliability of each channel throughout the video and is, therefore, equipped to handle varying conditions by online learning of channel weights. Unlike [112], our channel weight learning method provides fine-grained control over the impact of non-discriminative channels on tracker output. Moreover, the hyperparameter for channel selection ratio needs to be tuned for [50, 114], whereas our method dynamically determines the number of effective channels. Different from the reliability-based methods [55, 116] proposed previously, we use statistical measures to understand the strengths and limitations of our estimated reliability scores, thereby enhancing their interpretability. Also, unlike in [55, 116], we address the problem of false suppression of discriminative channels with low reliability scores by modeling inter-channel relations during channel weight learning. The authors of norm-based method in [115] were the first to introduce the idea of inter-channel similarity; however, their pointwise similarity in feature space is ineffective in higher dimensions, as shown by [117]. Our model measures the similarity between channels in terms of their target localization and assigns identical weights to discriminative channels. Furthermore, different from norm-based approach in [115], we use channel reliability scores (correlated with target localization accuracy) to compute channel weights. Also, [115] does not model the temporal consistency of learned weights.

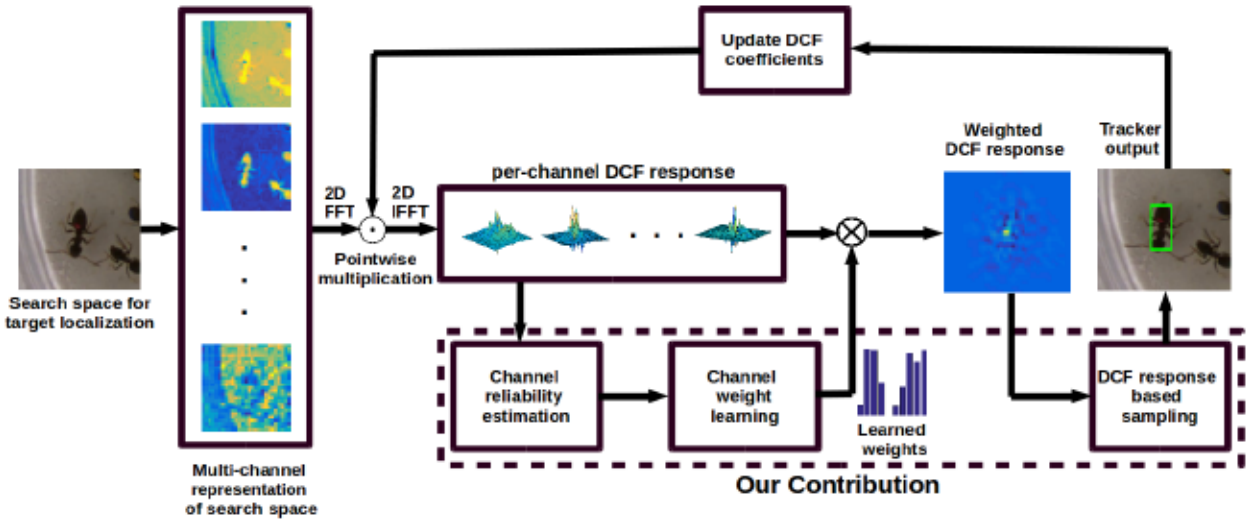

Figure 4.2: Pipeline of our reliability score-based channel-adaptive method for DCF tracking. In addition to channel reliability scores, our approach incorporates modeling of inter-channel relations and a temporal prior to compute the channel weights. The aggregated DCF response, obtained through a weighted sum of channel responses, is further analyzed for residual ambiguity in a post-processing (sampling) step.

In summary, prior work on channel adaptation either do not consider inter-channel relations [55, 116], do not estimate and use reliability scores [50, 112–115], or do not learn weights online [109–111]. We contribute a three-fold objective function that learns channel weights online depending on inter-channel relations, temporal prior, and per-channel reliability scores. This chapter extends our work in Chapter 3 by:

- A multi-scale pooling approach to estimate channel reliability scores based on local and global cues from per-channel filter responses. To better understand the adequacy of these estimated scores towards channel weight learning, we empirically verify their statistical relationship with target localization accuracy.

- A convex optimization problem to learn channel weights from reliability scores, coupled with temporal prior and an objective to model inter-channel relation. We also present a minimization algorithm to compute channel weights efficiently and sustain high tracking speed.

- A novel post-processing step (sparse sampling method) to refine target localization, resulting in reduced residual DCF response ambiguity after channel adaptation.

In addition, we integrate our method into both handcrafted and CNN-based DCF trackers. Figure 4.2 depicts the pipeline of the proposed channel weight learning method.

4.3 Proposed Channel Adaptation

4.3.1 Channel Reliability Estimation

In this section, we discuss our method to estimate channel reliability scores $r_i \in \mathbb{R}$ from per-channel responses C_i. Extending our idea of feature reliability estimation from Eq. 10 in Section 2.3.2, our method quantifies channel reliability r_i based on two complementary traits: margin to secondary (or distractor) peak $r_i^{(d)}$ and energy compactness $r_i^{(c)}$, to encapsulate local and global characteristics of channel response C_i, respectively. We compute these reliability scores by adopting a multi-scale pooling scheme, inspired by the popular scale-space framework in computer vision [118] and the stride size for pooling acts as a scale parameter.

Let n_j^{pool} indicate the stride size for pooling operation [33] at scale j. We use three scales in our implementation, i.e., $j = \{1, 2, 3\}$. Assuming the target object and search space dimensions are $D_i^o \times D_i^o$ and $D_i^z \times D_i^z$, respectively, in i^{th} channel representation (where $D_i^z > D_i^o$), we compute stride size n_j^{pool} as a function of D_i^z and D_i^o, i.e.,

$$n_j^{pool} = 2^{(j-1)} \left(\frac{D_i^z}{D_i^o} \right). \tag{19}$$

To compute the margin to distractor peak $r_{ij}^{(d)}$ for i^{th} channel response C_i at scale j, let P_{ij} denote the output of pooling operation on C_i, with stride size n_j^{pool}. Let (x_{p1}, y_{p1}) and (x_{p2}, y_{p2}) be the spatial locations of first and second largest values in P_{ij}, indicating the primary and secondary peaks, respectively. We use max-pooling [33] on C_i, since the operation preserves original values at primary and secondary peak. We define margin to distractor peak $r_{ij}^{(d)}$ as

$$r_{ij}^{(d)} = 1 - \frac{P_{ij}\left(x_{p2}, y_{p2}\right)}{P_{ij}\left(x_{p1}, y_{p1}\right)}. \tag{20}$$

Similar to Eq. 10 from Chapter 2, our method of quantifying channel reliability in Eq. 20 assigns higher score to channels whose responses have unique, stronger peaks compared to those having multiple peaks of similar strength.

The scoring method in Eq. 20 defines channel reliability based on the relative strength of primary and secondary peaks. However, our analysis suggests that a fraction of discriminative channels yields filter responses with weaker primary peaks, despite accurately locating the target object. Such behavior is possibly due to the presence of distractor entities in the vicinity of the target object or the global error reduction of least-squares [55] during the computation of filter coefficients H_i in Eq. 1, severely shrinking the value of the primary peak

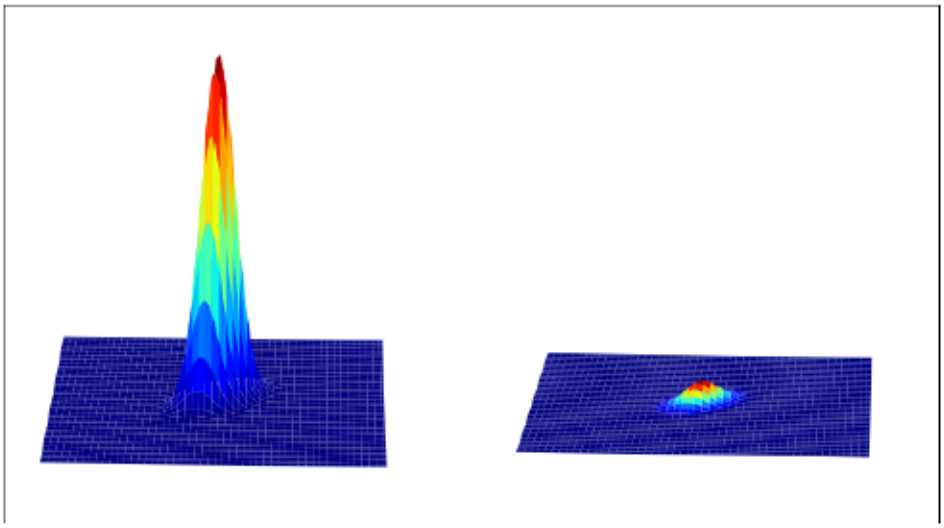

Figure 4.3: Visualizing the filter response of a channel at two far-apart frames in a video. We can see that, despite the filter response being unimodal, the peak value of the response varies significantly.

as depicted in Figure 4.3. These channels are incorrectly tagged to be non-discriminative due to low $r_{ij}^{(d)}$ score. Interestingly, despite the weakened primary peak, the energy of filter response in these channels is compact, i.e., concentrated in a smaller region around the primary peak. Therefore, the compactness of filter responses is a better metric to estimate the channel reliability scores in such cases. To measure such energy compactness, we propose to quantify the global behavior of channel responses as

$$r_{ij}^{(c)} = \frac{Q_{ij}(x_{max}, y_{max})}{\sum_{x}\sum_{y} Q_{ij}(x, y)}, \tag{21}$$

where Q_{ij} is the output of average pooling operation on energy map E_i with stride size n_j^{pool}, and (x_{max}, y_{max}) is spatial location of the largest value of Q_{ij} (we compute energy map E_i by element-wise squaring of C_i). Basically, $r_{ij}^{(c)}$ assigns high scores to channels with compact responses, and penalize those with distributed energy maps.

Finally, we fuse the reliability scores, $r_{ij}^{(d)}$ from Eq. 20 and $r_{ij}^{(c)}$ from Eq. 21, to compute the overall reliability score r_i for i^{th} channel as

$$r_i = min\left\{1, \quad \sum_{j=1}^{S} w_j r_{ij}^{(d)} + \sum_{j=1}^{S} w_j r_{ij}^{(c)}\right\}, \tag{22}$$

where w_j denote the non-negative weight assigned to estimates from scale j, and are chosen to sum upto 1. Our reliability scores r_i assigns higher scores (i.e., $r_i \approx 1$) to channels, whose

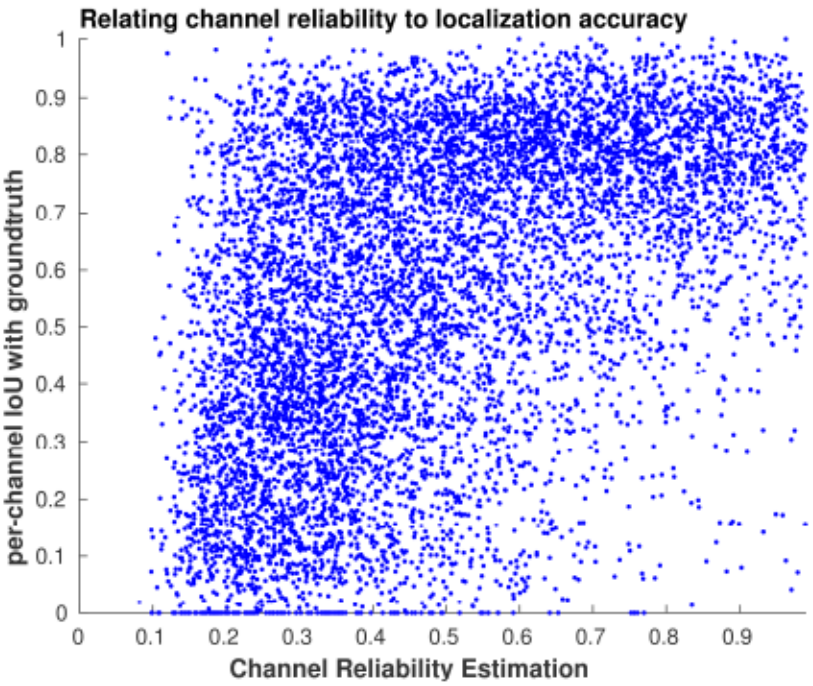

Figure 4.4: Visualizing joint distribution of per-channel reliability and corresponding IoU values for ECOHC [4] using OTB100 [6] dataset. We can see that reliability scores do not accurately reflect the target localization accuracy.

responses are either unimodal or energy compact (or both).

Before utilizing the reliability scores r_i in Eq. 22 to upweight the discriminative channels, it is essential to understand the extent to which r_i reflects the discriminative power of a channel. However, this is a challenging task since the discriminative power of a channel is an intangible metric. Hence, we achieve our goal by using per-channel localization accuracy as a proxy. The empirical results show that our channel reliability score r_i in Eq. 22 moderately correlates with per-channel localization accuracy, with Pearson correlation coefficient $\rho = 0.541$. Figure 4.4 visualizes the joint distribution of per-channel IoU and reliability scores from Eq. 22. We note the following observations from Figure 4.4:

- Our channel reliability scores do not precisely rank channels as their discriminative ability, reflecting the inherent noise in these estimated scores. To explain, a channel with IoU of 0.8 is assigned a reliability score between 0.5 to 0.9, as seen from Figure 4.4), which explains the weak correlation between the two metrics.

- Notably, a significant fraction of channels with higher localization accuracy is falsely assigned low reliability scores by our method (see the top-left area of Figure 4.4). It is because these mislabeled discriminative channels have indistinguishable filter responses

compared to non-discriminative ones and are assigned low weightage while computing the aggregated filter response.

Based on these observations, we propose a learning method to translate the noisy reliability scores r_i into meaningful channel weights, described in next Section 4.3.2 .

4.3.2 Channel Weight Learning

In this section, we propose a method to learn channel weights α_i from reliability scores r_i in Eq. 22, modeled as a convex optimization problem. This involves proposing regularizers, in the form of temporal constraints and inter-channel coupling to mitigate the impact of possible noisy channel reliability estimates. The proposed optimization problem for channel weights is,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{r}) + \gamma_1 \mathcal{T}(\boldsymbol{\alpha}, \boldsymbol{\beta}) + \gamma_2 \mathcal{I}(\boldsymbol{\alpha}), \\ s.t. \quad & 0 \leq \boldsymbol{\alpha} \leq 1, \end{aligned} \tag{23}$$

where $\{\boldsymbol{r}, \boldsymbol{\alpha}\} \in \mathbb{R}^N$ represent concatenated per-channel reliability scores and weights, respectively, and $\boldsymbol{\beta} \in \mathbb{R}^N$ is the temporal prior. Parameters $\gamma_1, \gamma_2 \in \mathbb{R}$ control the impact of regularizers $\mathcal{T}$ and $\mathcal{I}$, respectively.

The first term $\mathcal{L}$ in Eq. 23 increases the weightage of channels that consistently exhibit high reliability scores (and vice versa). The second term $\mathcal{T}$ controls the deviation of $\boldsymbol{\alpha}$ relative to temporal prior $\boldsymbol{\beta}$. The last term $\mathcal{I}$ models inter-channel relations, addressing the problem of false suppression of discriminative channels due to noisy reliability scores.

Loss Function $\mathcal{L}$

The key component towards learning effective channel weights is the loss function $\mathcal{L}$, based on reliability scores $\boldsymbol{r}$ as its input. The objective is to promote channels with higher reliability scores (i.e., $\alpha_i = 1$) and vice versa. We can see from Figure 4.3 that the threshold of 0.5 provides a good separation between discriminative and non-discriminative channels (except those on the left-top corner of the plot). As a first step towards designing the loss function $\mathcal{L}$, we categorize the channels into two sets, i.e., ones with $r_i \geq 0.5$ and the rest having $r_i < 0.5$. Finally, we aim to design a loss function such that it increases the weightage of i^{th} channel towards 1 if $r_i \geq 0.5$, and vice versa. Keeping these factors in mind, we define $\mathcal{L}$ as,

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{r}) = & \frac{1}{2} \sum_{i=1}^{N} \mathbb{1}_{r_i \geq 0.5} \cdot \max\left\{0, -(\alpha_i - r_i) + 1\right\}^2 \\ & + \frac{1}{2} \sum_{i=1}^{N} \mathbb{1}_{r_i < 0.5} \cdot \max\left\{0, (\alpha_i - r_i) + 1\right\}^2. \end{aligned} \tag{24}$$

The objective $\mathcal{L}$ is derived from the maximum-margin principle in Support Vector Machine classification to propel the learned weights α_i towards either 0 or 1, based on the channel reliability scores r_i.

Temporal Consistency Regularizer $\mathcal{T}$

The estimated channel reliability scores r_i in Eq. 22 may not accurately represent the discriminative power of a channel (see Figure 4.4). It could potentially introduce undesired temporal fluctuations in the learned weights α_i, leading to a non-smooth motion trajectory of the target object, thereby increasing the risk of tracker drift. To avoid such unwanted variations, we employ a temporal regularizer to enforce consistency of learned weights relative to $\boldsymbol{\beta}$ in Eq. 23. Prior $\boldsymbol{\beta}(t)$ at time t stores the weightage of channels in the past and is updated by a moving average scheme as follows,

$$\boldsymbol{\beta}(t) = \eta \cdot \boldsymbol{\beta}(t-1) + (1-\eta) \cdot \boldsymbol{\alpha}(t-1), \tag{25}$$

where η is the parameter to control the update rate.

For effective utilization of $\boldsymbol{\beta}$ during regularization, we also model the temporal behavior of per-channel discriminative power. This model is based on the statistical distribution of change in per-channel IoU between successive video frames, as depicted in Figure 4.5. It is evident from Figure 4.5 that the majority of changes are around zero implying there is a high probability that a channel retains its discriminative power across nearby frames. Figure 4.5 also indicates that there are sporadic instances of drastic changes between frames which is likely to happen during sudden scene changes (e.g., rapid target deformation). Accounting for such conceivable changes in per-channel discriminative power, we use *eps-*L_1 norm [119] as the temporal regularizer $\mathcal{T}$, i.e.,

$$\mathcal{T}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{N} \sqrt{(\alpha_i - \beta_i)^2 + \epsilon}, \tag{26}$$

where $\epsilon > 0$ is the associated parameter. As shown in Figure 4.6, the eps-L_1 norm provides the trade-off between L_2 and L_1 norms, stabilizing the learned weights against noisy variations while facilitating intermittent large changes.

Inter-Channel Coupling $\mathcal{I}$

As discussed in Section 4.3.1, the channel reliability estimator in Eq. 22 may misclassify a significant fraction of channels to be non-discriminative (*cf.* Figure 4.4). It leads to a scenario

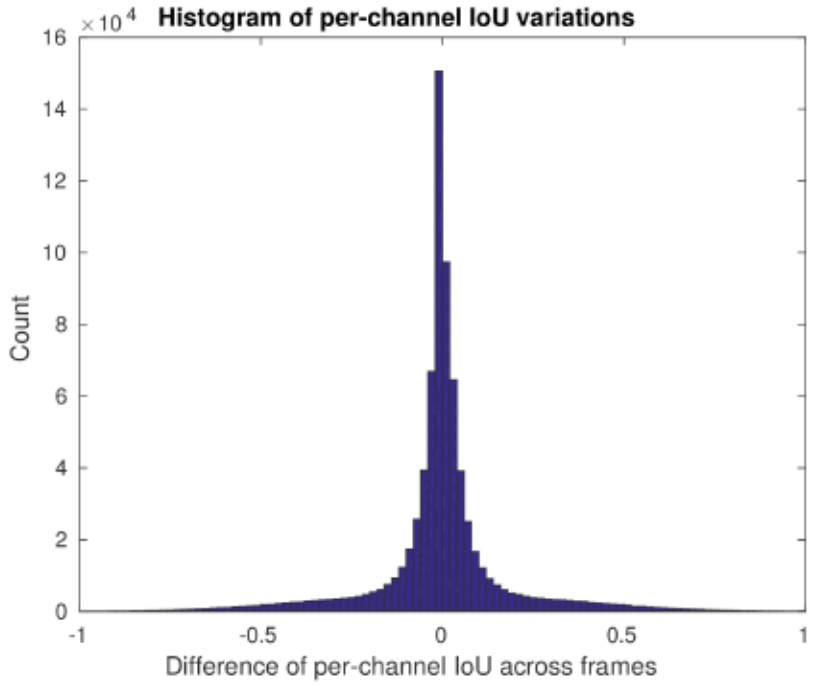

Figure 4.5: Histogram of per-channel IoU difference for ECOHC tracker, computed between successive frames of videos in OTB100 dataset. It summarizes temporal variations in per-channel discriminative power, providing insights to model an effective temporal regularizer.

where tracking output relies on few channels due to false suppression of discriminative ones. Since every channel encodes partial information about the target, determining the tracker output based only on a few channels (say, less than 25%) introduces tracking errors. To avoid such false suppression of discriminative channels, we propose objective $\mathcal{I}$ in Eq. 23. As a prerequisite to $\mathcal{I}$, we define a pairwise similarity metric s_{il} between i^{th} and l^{th} channels as,

$$s_{il} = \begin{cases} 1, & \text{if IoU}(i,l) \geq 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

Eq. 27 states that the two channels are similar (in terms of reliability) if their bounding box (BB) outputs have IoU ≥ 0.5. Based on this pairwise similarity metric, we define $\mathcal{I}$ as,

$$\mathcal{I}(\alpha, r) = \sum_{l=1}^{N} \frac{1}{N_h} \left(\sum_{i=1}^{N} \mathbb{1}_{r_i \geq 0.5} \cdot s_{il} \cdot \max\left\{0, \alpha_i - \alpha_l\right\}^2 \right), \tag{28}$$

where N_h is the total number of channels with $r_i > 0.5$. To comprehend the functionality of proposed objective $\mathcal{I}$ in Eq. 28, let us assume that the i^{th} channel succeeds in locating

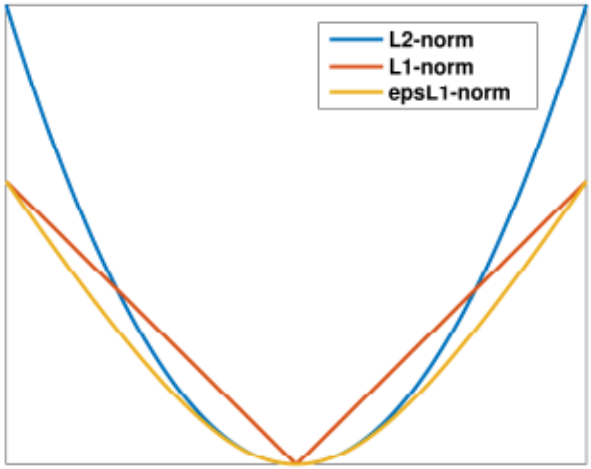

Figure 4.6: Visualizing the behavior of regularizer norm functions. We can see that eps-L_1, with $\epsilon = 1$, shares properties of L_1 and L_2 norm, therefore better suited for effective suppression of non-discriminative channels.

the target object and assigned a high reliability score (i.e., $r_i > 0.5$). On the contrary, let channel l has a low reliability score (i.e., $r_l < 0.5$) despite accurately localizing the target object. Here, the BB output of i^{th} and l^{th} channels have significant overlap, since they are localizing the same target. In this case, objective $\mathcal{I}$ penalizes the difference in learned weights for channels i and l, thereby preventing false suppression of channel l. The inner summation in Eq. 28 considers channels with high reliability scores as anchors, impacting the weight of others. If two channels have negligible overlap between their BB output, s_{il} will be zero, and (28) has no effect on their learned weights. Figure 4.7 visualizes the effect of the proposed $\mathcal{I}$ on computed weights.

4.3.3 Solving for Channel Weights

This section presents our approach to solving the optimization problem in Eq. 23 for channel weights. It should be noted that Eq. 23 has no closed-form solution, and using external solvers such as CVX [102] significantly affects the tracking speed. Therefore, we propose an algorithm based on the alternating minimization method [104] to iteratively solve for channel weights with minimal impact on tracking speed. We leverage the constraints in Eq. 23, which are separable and variable-bounded (also known as, box constraints), to facilitate faster convergence of learned weights using the bisection method [104].

To briefly explain the working of the proposed Algorithm 3, let $O(\boldsymbol{\alpha})$ be the value of the objective function in Eq. 23 for a given $\boldsymbol{\alpha}$ and $\partial O(\alpha_i)$ be the subgradient of $O(\boldsymbol{\alpha})$

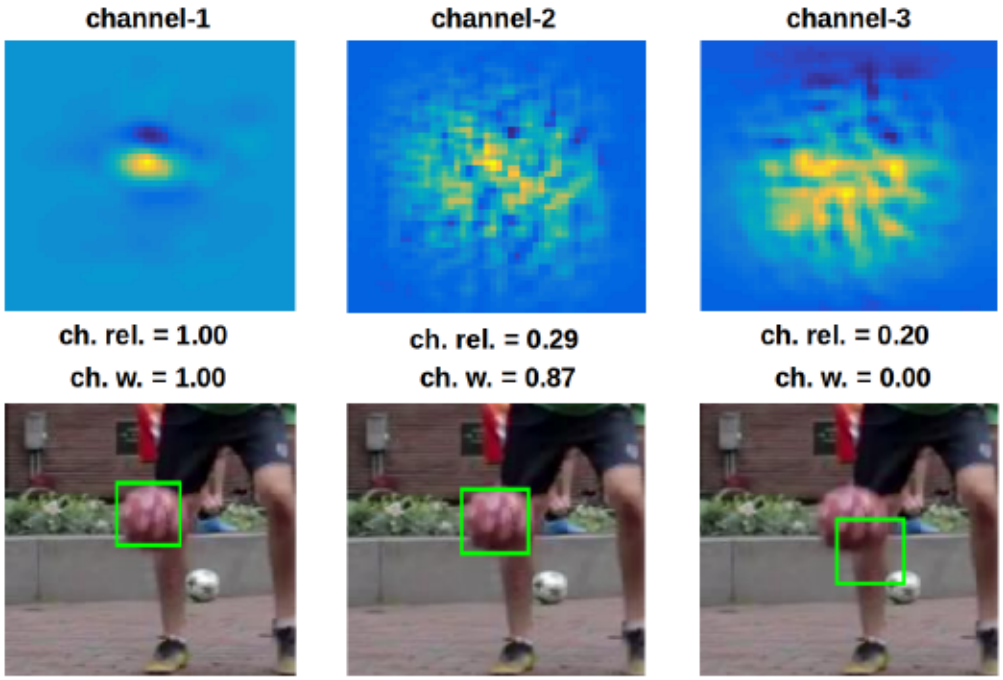

Figure 4.7: Visualizing channel responses (top row), reliability scores and channel weights (center row), and corresponding target localization results (last row). Despite having a low reliability score, channel-2 is assigned higher weight because of its significant IoU value with highly reliable channel-1. Channel-3 is assigned low weight, since it neither has high reliability score nor significant overlap with the bounding box output of channel-1.

with respect to α_i. We begin our algorithm by iterating over the channel weights, whose range is set between 0 and 1 as per the constraint in Eq. 23. We then recursively shrink the feasible region by half until we observe no change in the value of i^{th} channel weight (*line-12*). After performing these steps for all the channel weights, we get an optimal weight vector at iteration number v. Finally, we measure the difference in ℓ_2 norm of the weight vectors from the current and previous iteration (*line-21* of Algorithm 3). If the computed value of norm is less than a threshold δ, we declare that our algorithm has converged. For the iteration over each channel, we set the maximum number of iterations, $NumIter = 10$ in Algorithm 3, based on the following convergence criterion [104],

$$\left\| \alpha_i^* - \alpha_i^{(u)} \right\| \leq \frac{1}{2^u}, \tag{29}$$

where α_i^* be the optimal weight for the i^{th} channel, and $\alpha_i^{(u)}$ be the channel weight obtained after u iterations. Similarly, we set the value of $\delta = 10^{-3}$ to check the convergence of our algorithm. Our experimental results indicate that, with the proposed Algorithm 3, we obtain the desired channel weights with sufficient precision while maintaining high tracking speed.

Input: channel reliability scores: $\boldsymbol{r}$ and prior weights: $\boldsymbol{\beta}$

while *True* **do**
  for $i : 1$ *to* N **do**
    Initialize: $u = 0$, $\alpha_- = 0$, $\alpha_+ = 1$
    while $u \leq NumIter$ **do**
      $\alpha_i^{(u)} = \dfrac{\alpha_- + \alpha_+}{2}$
      if $\partial O(\alpha_i^{(u)}) > 0$ **then**
        $\alpha_- = \alpha_i^{(u)}$
      end
      if $\partial O(\alpha_i^{(u)}) < 0$ **then**
        $\alpha_+ = \alpha_i^{(u)}$
      end
      if $\partial O(\alpha_i^{(u)}) = 0$ ***or*** $(\alpha_+ - \alpha_-) < \delta$ **then**
        break
      end
      $u \rightarrow u + 1$
    end
    $\alpha_i^{opt} = \underset{idx=1,\ldots,u}{\arg\min}\, O(\alpha_i^{(idx)})$
  end
  $v \rightarrow v + 1$
  $\boldsymbol{\alpha}(v) = \left[\alpha_1^{opt}, \cdots, \alpha_N^{opt}\right]$
  if $\|\boldsymbol{\alpha}(v) - \boldsymbol{\alpha}(v-1)\|_2 < \delta$ **then**
    break
  end
end

Output: channel weights: $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_N\}$

Algorithm 3: Alternating minimization algorithm to learn channel weights using the bisection method.

4.3.4 Sampling the DCF Response

In this section, we propose a post-processing step by sampling the aggregated DCF response to refine target localization (*cf.* Figure 4.2). Even though the proposed sampling method is a non-integral part of our channel weight adaptation, this computationally inexpensive method reduces the chances of tracker drift. Also, it can be easily integrated into any DCF tracker using two or more features.

The occurrence of a challenging condition (such as fast motion) during tracking affects the discriminative power of channels differently at each frame. Based on its severity, only a limited percentage of channels succeed in precisely locating the target object (say around

10%). The abundance of non-discriminative channels causes residual ambiguity in the overall DCF response despite using adaptive weights. It could lead to tracker drift since the location of the global maximum in the final filter response may not correspond to the target location. To alleviate this problem, we re-examine prominent peaks in the aggregated filter response and consider these peaks as initial estimates (candidates) for the target center. We find these candidate locations by performing 3×3 non-maximal suppression on the overall DCF response. Next, we extract features from a rectangular search space around these locations and perform correlation with the filter coefficients H_i from Eq. 1. From the filter responses computed from candidate locations, we select the one with the highest response value as the target center. We initiate this sampling method when we observe a discrepancy between bounding box outputs from feature responses by defining $\{F_1, F_2, ..., F_M\}$ as filter responses corresponding to M features utilized by the DCF tracker. We compute the filter response corresponding to the k^{th} feature, F_k, as

$$F_k = \sum_{i=1}^{N} \mathbf{1}_{i \in k} \cdot \alpha_i C_i, \tag{30}$$

where C_i is the per-channel response, α_i is the learned channel weight and $\mathbf{1}_{i \in k}$ is the indicator function defined as,

$$\mathbf{1}_{i \in k} = \begin{cases} 1, & \text{if } i^{th} \text{ channel belongs to } k^{th} \text{ feature} \\ 0, & \text{otherwise.} \end{cases}$$

Let $\{B_{F_1}, B_{F_2}, ..., B_{F_M}\}$ be BB outputs generated based on these feature responses. If $\text{IoU}(B_{F_1}, ..., B_{F_M}) < 0.25$ then we initiate the sampling method.

4.4 Experimental Results and Analysis

We begin this section by presenting our simulation setup. Then, we objectively compare the performance of the proposed method with the baseline and related trackers, followed by an ablation study and per-attribute analysis.

4.4.1 Setup

To compare the performance of the proposed method with other trackers, we use two evaluation measures: Average Overlap Ratio (AOR) for accuracy and Failure Rate (FR) for robustness. As explained in Section 1.3, AOR measures the extent of overlap between

groundtruth and the predicted bounding box averaged over the dataset. Similarly, FR computes the average fraction of frames with predicted bounding boxes having zero overlap with the groundtruth. Therefore, a tracker with high-quality performance exhibits high AOR and low FR. Different than VOT [5, 120], we do not use the Expected Average Overlap (EAO) metric with a reset protocol. It is because not all trackers that encounter drift remain off the target till the end of a sequence, as assumed by [5] and [120]. Many trackers [27, 67] recover after short-term failure, and there are trackers such as [71], employing explicit mechanisms for failure recovery. Therefore, we follow the no-reset protocol, i.e., a tracker is not reset upon drifting off the target.

We test our dynamic channel-adaptive (*dca*) method by integrating it into two prominent DCF trackers using CNN features, i.e., ECO [4] and DeepSTRCF [16]. ECO deploys a combination of handcrafted (i.e., HOG and CN templates) and deep features (i.e., output from *conv-1* and *conv-5* layers of pretrained VGG-M [17] network). Similarly, DeepSTRCF deploys the same handcrafted features as ECO, but it uses the output of the VGG-M *conv-3* layer as deep features. For a fair comparison with the baseline, we use the same feature sets in our experiments. We use the datasets OTB100 [6] (100 video sequences), TC128 [93] (128 sequences), and short-term VOT2018 [5] (60 sequences) in our simulation.

Parameters of the proposed *dca* method are w_j of Eq. 22, γ_1 and γ_2 of Eq. 23, ϵ of Eq. 26, and η in Eq. 25. We use OTB100 [6] dataset (100 test sequences) to choose optimal values of these parameters and set the same values $\{w_1 = 4/7; w_2 = 2/7; w_3 = 1/7\}$ and $\epsilon = 1$ for both *dca*-ECO and *dca*-DeepSTRCF, while we assign $\gamma_1 = \{40, 2\}$, $\gamma_2 = \{25, 15\}$, and $\eta = \{1/20, 1/16\}$, respectively, for *dca*-ECO and *dca*-DeepSTRCF. With these parameters tuned using OTB100 dataset, we observe comparable AOR but an average improvement of 17.7% in terms of FR across *dca*-ECO and *dca*-DeepSTRCF. We compute channel weights every five frames to maintain high tracking speed. It also helps computing better estimates of pairwise metric s_{il} from Eq. 27 by averaging over multiple frames, thereby avoiding overfitting to single-frame estimates. Channel weights at the first frame were initialized to 1, i.e., $\alpha_i(0) = 1$.

4.4.2 Comparison to Baseline and Related Trackers

In this section, we evaluate our *dca* method when integrated into the baseline trackers using CNN features, i.e., ECO [4] and DeepSTRCF [16]; and those using handcrafted features, ECOHC [4] and STRCF [16]. In addition, we compare to channel-adaptive CGRCF [115] tracker and channel-selective trackers, ACSDCF [114], and GFSDCF [50]; these trackers use CNN features. We evaluate all trackers with code and parameters provided by their authors.

Trackers	TC128		VOT2018		Aggregated Dataset				
	AOR↑	FR↓	AOR↑	FR↓	AOR↑	improvement	FR↓	improvement	fps
dca-DeepSTRCF (ours)	0.6099	0.1211	0.3997	0.2476	0.5431	2.9%	0.1612	14.5%	1.6
DeepSTRCF [16] (CVPR'18)	0.5955	0.1472	0.3819	0.2778	0.5277	-	0.1886	-	2
dca-ECO (ours)	0.5999	0.1091	0.3786	0.2467	0.5297	2.0%	0.1528	12.7%	7
ECO [16] (CVPR'18)	0.5936	0.1159	0.3598	0.3026	0.5194	-	0.1751	-	12
CGRCF [115] (TCSVT'21)	0.5671	0.1610	0.3777	0.2986	0.5070	4.5%	0.2047	25.3%	10
ACSDCF [114] (IJCV'21)	0.5303	0.1439	0.3919	0.2217	0.4864	8.9%	0.1686	9.4%	3
GFSDCF [50] (ICCV'19)	0.5501	0.1477	0.3840	0.2596	0.4974	6.5%	0.1832	16.6%	2

Table 4.1: Proposed *dca* versus baseline CNN-based ECO [4] and STRCF [16], and norm-based CGRCF, ACSDCF, and GFSDCF under TC128 and VOT2018, where all the trackers use VGG [17] features. The best results are highlighted in red and second best in green. The "improvement" of proposed *dca* is given with respect to baselines, while "improvement" against CGRCF, ACSDCF, and GFSDCF are with reference to *dca*-ECO.

Since we used OTB100 for training, i.e., fine-tuning the parameters, we validate our method with TC128 and VOT2018. We use the original bounding boxes as groundtruth during evaluation (i.e., polygonal for VOT2018 and rectangular for TC128). Trackers typically perform differently between datasets. To test their generalization ability, we combined individual test datasets TC128 and VOT2018, to calculate AOR and FR values on the aggregated dataset.

We conclude from Table 4.1 that the proposed *dca* improves the performance of CNN-based baseline trackers on average by 13.6% in FR and 2.45% in AOR. Compared to the channel-adaptive CGRCF [115], proposed *dca*-ECO has a lower FR by 25.3% and higher AOR by 4.5% on the aggregated dataset. Our method also performs better than channel-selective trackers: ACSDCF (*dca*-ECO has 9.4% lower FR and 8.9% higher AOR) and GFSDCF (*dca*-ECO has 16.6% lower FR and 6.5% higher AOR) on the aggregated dataset. We note that higher improvement is achieved in terms of FR, indicating the usefulness of the proposed method in reducing tracking failures. Note that to ensure a fair comparison, we use VGG [17] features for all CNN-based DCF trackers ECO, DeepSTRCF, CGRCF, ACSDCF, and GFSDCF. The channel-adaptive JCRCF [116] is excluded since its authors did not release the tracker code. The *fps* averaged over all videos of a dataset is given in the last column of Table 4.1; using the proposed *dca* method, the *fps* decreases on average by 30% compared to baseline. Our simulations were conducted on Nvidia RTX8000 GPU.

Furthermore, Table 4.2 presents the results of the proposed *dca* integrated into two DCF trackers using handcrafted features, i.e., ECOHC [4] and STRCF [16]. We can see that our method improves STRCF by 17.3% in terms of FR and 4.5% in terms of AOR, while improving ECOHC by 12.2% in terms of FR and 4.2% in terms of AOR on the aggregated dataset.

Trackers	TC128		VOT2018		Aggregated Dataset				
	AOR↑	FR↓	AOR↑	FR↓	AOR↑	improvement	FR↓	improvement	fps
dca-STRCF (ours)	0.5778	0.1386	0.3678	0.3070	0.5111	4.5%	0.1920	17.3%	15
STRCF [16] (CVPR'18)	0.5551	0.1683	0.3518	0.3267	0.4906	-	0.2186	-	20
dca-ECOHC (ours)	0.5715	0.1342	0.3524	0.3251	0.5019	4.2%	0.1948	12.2%	40
ECOHC [4] (CVPR'17)	0.5512	0.1694	0.3272	0.3777	0.4801	-	0.2355	-	50

Table 4.2: Proposed *dca vs* baseline trackers with hard-crafted features, ECOHC [4] and STRCF [16] on TC128 and VOT2018. The best results are in red and second best in green.

Trackers	AOR↑	improvement	FR↓	improvement
DeepSTRCF [16] baseline	0.5827	-NA-	0.1483	-NA-
ℓ_1 norm	0.5852	0.43%	0.1427	3.78%
ℓ_2 norm	0.5883	0.96%	0.1351	8.90%
eps-L_1 norm	0.5958	2.25%	0.1219	17.80%

Table 4.3: Comparing *dca* results using different temporal regularizers on the aggregated dataset (OTB100, TC128, and VOT2018).

4.4.3 Ablation Study

In this section, we run the ablation study for *dca*-DeepSTRCF under the aggregated dataset consisting of all 289 sequences from OTB100, TC128, and VOT2018.

Role of Temporal Regularizer Temporal constraint $\mathcal{T}$ on channel weights in Eq. 23 is essential to alleviate the impact of noisy reliability scores. Table 4.3 summarizes tracker results for different choices of temporal regularization. We argue that it is important to choose an appropriate regularizer to maximize the benefits of our method. We empirically verify this by incorporating ℓ_1 and ℓ_2 norms into our channel adaptation method to evaluate their effectiveness, as shown in rows 2 and 3 of Table 4.3, respectively. We observe that the proposed eps-L_1 norm achieves superior results among other regularizers, with a two-fold relative improvement in FR compared to the second best ℓ_2 regularizer.

Role of Inter-Channel Coupling and Sampling By excluding the proposed inter-channel coupling objective $\mathcal{I}$ in Eq. 23 and sampling in Section 4.3.4, we observe only a marginal improvement of 0.88% in FR compared to the baseline, as shown in second row of Table 4.4. With the addition of inter-channel coupling term $\mathcal{I}$ in Eq. 23, our method avoids false suppression of discriminative channels. As a result, we observe a significant improvement of 15.98% in FR, as shown in third row of Table 4.4. Finally, we obtain the best results by incorporating our sampling method, as shown in the last row of Table 4.4, demonstrating its role in handling residual ambiguity in the overall DCF response.

Tracker	$\mathcal{I}$ from Eq. 23	Sampling method from Section 4.3.4	AOR↑	improvement	FR↓	improvement
DeepSTRCF [16] baseline	-	-	0.5827	-NA-	0.1483	-NA-
dca-DeepSTRCF	✗	✗	0.5779	-0.82%	0.1470	0.88%
dca-DeepSTRCF	✓	✗	0.5939	1.92%	0.1246	15.98%
dca-DeepSTRCF	✓	✓	0.5958	2.25%	0.1219	17.80%

Table 4.4: Ablation study results for *dca* method on the aggregated dataset (OTB100, TC128, and VOT2018).

4.4.4 Per-Attribute Analysis

We analyzed the FR performance of *dca*-DeepSTRCF on 289 videos aggregated across three datasets (OTB100, TC128, and VOT2018) for 11 challenging attributes: background clutter, deformation, fast motion, illumination variation, in-plane rotation, low resolution, motion blur, occlusion, out-of-plane rotation, out-of-view, and scale variation. We observed variable improvement in FR under the single attributes: 34.8% for in-plane rotation (108 videos), 33.8% for background clutter (74 videos), 32.8% for illumination variation (91 videos), 29.3% for out-of-plane rotation (137 videos), 27.1% for deformation (80 videos), 24.3% for occlusion (142 videos), 21.5% for scale variation (162 videos), 19.1% for low resolution (26 videos), 19% for motion blur (66 videos), 15.4% for fast motion (141 videos), and 10.8% for out-of-view (26 videos). The average improvement for overall attributes is 17.80%.

Figure 4.8 visualizes samples of qualitative tracker results under challenging attributes: in-out-of-plane rotation (*girl*), fast motion (*ants1*), background clutter (*Railwaystation*), and multi-attributes with deformation, scale, and variable illumination (*Skiing*). We can see how proposed *dca* corrects baseline tracker's failures in these videos.

4.5 Summary

The inherent challenges in multi-channel DCF tracking, particularly related to fixed channel weights and the impact of unreliable channels on tracker results, prompted our exploration of channel reliability-based approaches. Our investigation revealed a notable issue – a significant proportion of discriminative channels received erroneously low reliability scores. This misclassification led to the unwarranted suppression of discriminative channels, resulting in a degradation of overall tracking performance. The similarity in filter responses between these misclassified and non-discriminative channels posed a unique challenge for rectification based solely on individual channel responses. In response to these challenges, we introduced a novel method that incorporates inter-channel coupling to learn channel weights from noisy reliability estimates. It also integrates a regularizer to ensure temporal consistency of these

Figure 4.8: Sample results showing reduced tracker failures when using the proposed *dca*. The color coded bounding boxes represent groundtruth labels, output of base trackers: ECO and DeepSTRCF, and proposed channel-adaptive trackers: *dca*-ECO and *dca*-DeepSTRCF.

weights and employs a sampling method to address the impact of residual ambiguity in the aggregated filter response on target localization.

We empirically verified the effectiveness of our approach by integrating the proposed dynamic channel-adaptive (*dca*) method into DCF trackers with handcrafted and CNN features. Experimental results on three datasets (OTB100, TC128, and VOT2018) show that our method improves the baseline trackers ECO and STRCF; for example, tracking drift, indicated by FR, of ECO was reduced by 12.7% compared to baseline. Furthermore, in comparison to related CNN-based trackers – specifically, the channel-adaptive CGRCF and channel-selective ACSDCF and GFSDCF – our *dca*-ECO achieved an average improvement of 17.1% in terms of FR.

The relevance of our channel adaptation methodology extends to trackers designed for low-memory embedded devices, which often utilize features from lightweight backbones [121, 122]. In such resource-constrained scenarios, channel-adaptive methods can compensate for the absence of powerful features from deeper models, ultimately enhancing tracker

performance. Additionally, the proposed method has a low memory footprint since it does not involve any intensive matrix-to-matrix multiplications.

A limitation inherent in the broader DCF-based approach is that it relies on solving a large scale convex optimization problem to learn the filter coefficients (*cf.* Eq. 1). Specific to our channel-adaptive method for multi-channel DCF tracking, seamless integration into an end-to-end trainable architecture is challenging, primarily due to the iterative nature of the proposed Algorithm 3. Addressing these limiations involve a broader research problem of constructing generic convex optimization solvers, that are capable of effectively harnessing the parallel processing capabilities of GPU infrastructure. Furthermore, DCF framework lack the flexibility in leveraging large-scale external datasets through an offline training phase, a strategy that has proven effective in enhancing tracking performance, particularly for Siamese Network-based (SN) trackers. Unlike DCF trackers, the SN architecture supports end-to-end training and inference on GPUs, hence they can fully exploit the computational capabilities of the specialized hardware. Due to these advantages of SN paradigm over the DCFs, in the next two chapters, we investigate SN-based approach for VOT.

Chapter 5

Mobile Vision Transformer-based Visual Object Tracking

5.1 Introduction

In this chapter, which is based on our paper [97], we broaden our exploration of efficient visual object tracking (VOT) by delving into Siamese Network-based (SN) models. While Discriminative Correlation Filter-based (DCF) trackers excel in robustness against semantic background by explicitly learning target-specific filter coefficients, the landscape of VOT has evolved with the emergence of large-scale training datasets and the adoption of powerful backbones for robust feature representation. In recent years, SN trackers have demonstrated state-of-the-art performance across various benchmarks [8–10]. Notably, SNs exhibit higher inference speed compared to DCF trackers, facilitated by their model architecture supporting end-to-end testing on a GPU.

The SN architecture consists of a backbone to generate feature representation of the target template and search regions, a localization head module for target state estimation, and an optional feature fusion module for relation modeling (*cf.* Figure 1.4). Among these components, the backbone plays a crucial role in determining the accuracy and robuestness of a tracking algorithm [123]. Most SN trackers use the ResNet [35] backbone for feature extraction, with ResNet-50 and ResNet-101 being popular choices. The more recent trackers use pre-trained Vision Transformer models [84,124] as their backbone, surpassing the performance of their ResNet-based predecessors. These fully transformer-based tracking methods have unified feature extraction and relation modeling by deploying self and mixed attention blocks in their backbone [13,15] to simplify the SN architecture further. However, a notable disadvantage of these fully transformer-based trackers is the complexity of their backbone,

both in terms of memory (a large number of model parameters) and latency (low inference speed). By deploying these models, achieving high tracking speed on an ordinary CPU or mobile device is challenging. This limitation severely restricts the usage of such tracking algorithms for several resource-constrained applications.

On the other hand, most lightweight tracking algorithms employ compact Convolutional Neural Network-based (CNN) backbones to minimize the model latency. The inductive biases of convolutional blocks effectively model the spatially local information related to the target object but fail to capture the global relations essential for accurate target state estimation in tracking [13]. Such a lack of global association between the template and search region in the backbone increases the burden on the feature fusion module (or the neck) to generate the fused encoding favorable for accurate tracking. Deploying self-attention-based transformers [2] in the backbone has been shown to be effective at global contextual modeling and excellent for tracking [14, 15]; however, these fully transformer-based backbones are computationally expensive.

In this chapter, we explore the potential of Mobile Vision Transformers (MobileViTs) as the backbone for single object tracking. Recent MobileViTs, equipped with a cascade of CNN and transformer blocks, are known for their lightweight architecture, low latency, and versatility across tasks such as object detection and semantic segmentation [125]. Building upon these advancements, we propose a lightweight yet high-performance Mobile Vision Transformer-based tracking algorithm, *MVT*. Moreover, unlike conventional lightweight trackers that independently process template and search region features in their respective backbones, our *MVT* algorithm introduces a hybrid feature extraction method, which blends the target template and search regions within the backbone using our novel Siamese Mobile Vision Transformer (Siam-MoViT) block.

5.2 Related Work

Multiple SN lightweight trackers have been presented in the last few years. LightTrack [121] employed Neural Architecture Search [126] to present an efficient tracking pipeline. It designed a search space of lightweight building blocks to find the optimal backbone and head architectures with pre-set constraints on the number of model parameters. E.T.Track [127] proposed Exemplar Transformers for tracking to achieve real-time speed on a CPU. It used a stack of lightweight transformer blocks in the head module for target classification and bounding box regression. FEAR [122] tracker deployed a dual-template representation to incorporate temporal information during tracking. With a compact backbone, FEAR achieved over 200 frames-per-second (*fps*) speed on iPhone 11 with negligible impact on

battery level. Stark-Lightning [13] used a RepVGG [128] backbone and a transformer-based encoder-decoder architecture in the neck module to model spatio-temporal feature dependencies between the target template and search regions. HiFT [129] proposed a hierarchical feature-based approach for aerial tracking. It generated hierarchical similarity maps from the multi-level convolutional layers in the backbone and fused the shallow and deep features using transformers. SiamHFFT [130] extended the hierarchical feature fusion approach by [129] to model the inter-dependencies within the multi-level features and achieve high tracking speed on a CPU.

Among the related lightweight trackers, LightTrack is closest to our work, having similar neck and head modules but a different backbone. Stark-Lightning uses a transformer-based neck module to fuse features from the template and search regions. In contrast, the proposed *MVT* uses a simple, parameter-free cross-correlation operation in its neck module. As a post-processing step, the related trackers LightTrack and E.T.Track refine their predicted bounding boxes by penalizing significant changes in bounding box size and aspect ratio between consecutive frames. Unlike these trackers, the proposed *MVT* does not perform such heuristic-driven bounding box refinements.

Most importantly, all the related lightweight trackers use a two-stream approach during feature extraction, i.e., the backbone features from the template and search region are computed independently. Such a two-stream computation limits the interaction between the template and search regions to the neck module only, resulting in inferior tracking performance. To alleviate this problem, we propose a hybrid feature extraction method where template and search regions are blended in the backbone by our novel Siam-MoViT block, as shown in Figure 5.1. The resulting entangled feature representation generated using our Siam-MoViT block improves the tracker performance while maintaining high inference speed. Efficient transformer architectures is an emerging research topic [3] and has been unexplored by previously proposed lightweight trackers. To our knowledge, we are the first to use MobileViT as the backbone for object tracking.

Our contributions are:

- A novel lightweight tracking algorithm using MobileViTs. We show that the proposed *MVT* tracker performs better than related lightweight trackers.
- A hybrid feature extraction approach, intertwining the template and search regions using our Siam-MoViT block, producing better features for target state estimation.

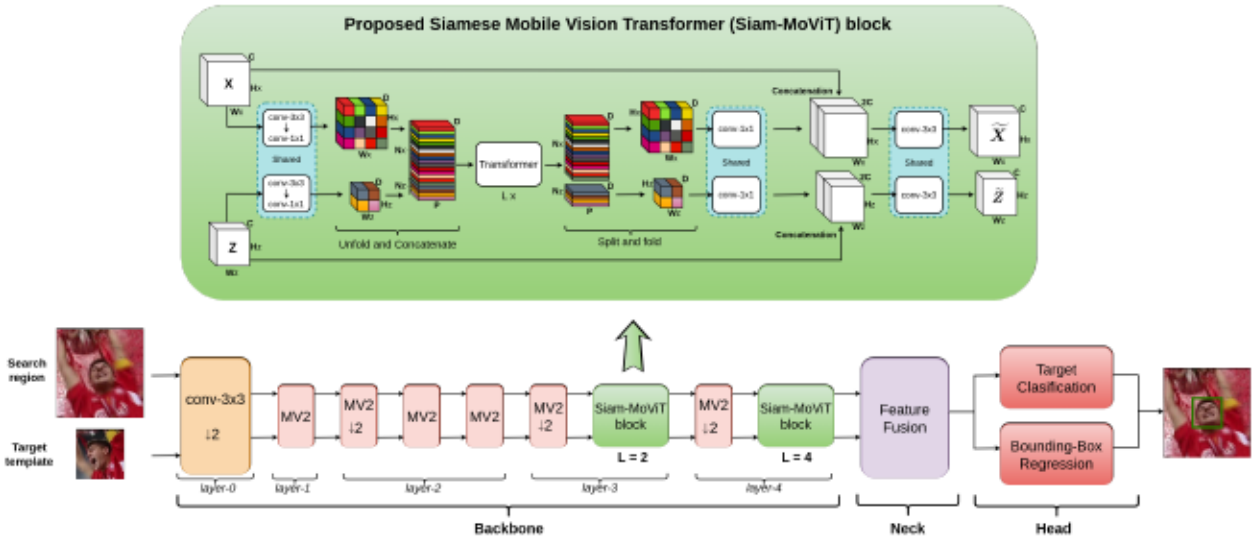

Figure 5.1: The pipeline of the proposed *MVT* tracker and our Siam-MoViT block. The backbone consists of MobileNetV2 [7] (or MV2) and Siam-MoViT blocks for feature extraction. ↓2 indicates spatial downsampling by a factor of 2. Details of our Siam-MoViT block can be found in Section 5.3.

5.3 Proposed Mobile Vision Transformer-based Tracker

In this section, we discuss the pipeline of our *MVT* algorithm for single object tracking (shown in Figure 5.1) and information related to model training.

5.3.1 Proposed *MVT* Backbone and the Siam-MoViT block

The input to our *MVT* backbone is a pair of the target template and search region image patches, $Z_{in} \in R^{W_z \times H_z \times 3}$ and $X_{in} \in R^{W_x \times H_x \times 3}$, respectively. The tracker backbone consists of cascaded MobileNetV2 [7] and the proposed Siam-MoViT blocks, as shown in Figure 5.1. These modules process the input image patches sequentially, with recurring spatial down-sampling operations to reduce the feature dimensionality. The proposed Siame-MoViT block uses a modified MobileViT block [125], especially around the transformer block, to accommodate features from the template and search region.

Our Siam-MoViT block receives a pair of intermediate feature maps Z and X, belonging to the template and search regions, respectively. We assume that Z and X have C channels. Inside the Siam-MoViT block, first, we apply a 3×3 convolutional filter to learn spatially local feature representations. It is followed by a 1×1 convolutional filter, projecting the features onto a D-dimensional space as a linear combination of C input channels. Next, we perform the *unfold and concatenate* operation (*cf.* Figure 5.1), where we divide the feature maps X and Z into N non-overlapping patches of size $w \times h$. We then flatten these patches

to generate tokens of size $P \times N \times D$, where $P = w \cdot h$ and $N = \frac{W \cdot H}{P}$. These tokens are concatenated and passed through a series of L multi-headed self-attention transformer blocks to encode the global relationship between the template and search regions [2]. This operation of learning self-attention on the concatenated features facilitates the exchange of information between template and search regions, thereby generating high-quality encodings for accurate target localization. To restore the spatial ordering of feature maps, we split the output tokens from the transformer and re-arrange them to obtain feature maps of size $H_z \times W_z \times D$ and $H_x \times W_x \times D$, shown as the *split and fold* operation in Figure 5.1. Then, we re-map the number of channels from D to C by applying a 1×1 convolutional filter and concatenate the resulting feature maps with the inputs to the Siam-MoViT block, i.e., Z and X. Finally, we apply a 3×3 convolutional filter on the concatenated feature maps to generate the output of our Siam-MoViT block, denoted as $\tilde{Z}$ and $\tilde{X}$, having the same size as Z and X, respectively. Note that all the MobileNetV2 blocks in the backbone and the CNN blocks within the Siam-MoViT module are applied separately to template and search regions, as shown in Figure 5.1, with shared weights.

5.3.2 Neck and Head Modules

The output from the last layer of the *MVT* backbone has feature maps corresponding to the template and search region. We fuse these features in the neck module to generate an encoded feature representation $f_{zx} \in R^{\frac{H_z \cdot W_z}{16^2} \times \frac{H_x}{16} \times \frac{H_x}{16}}$. For this feature fusion, we use a simple pointwise cross-correlation operator [63] in the neck module, same as LightTrack [121]. We use a layer of batch-normalization (BN) [131] before performing cross-correlation. We then apply a 1×1 convolutional *channel-adjust* layer on f_{zx} to match the number of channels between f_{zx} and the head module.

For target classification and bounding-box regression, we adopt the head module from [14], which uses a fully convolutional network (FCN) to perform classification and regression. The FCN consists of a stack of five Conv-BN-ReLU blocks. The classification network predicts a score map $\mathcal{R} \in R^{\frac{H_x}{16} \times \frac{W_x}{16}}$, and the location of the maximum value in $\mathcal{R}$ is considered as the target location. The regressor network predicts the normalized bounding box size (i.e., target width and height) and corresponding local offset values.

5.3.3 Loss Function for Training

During training, we use loss functions for the classification and regression output by the head module of our *MVT* tracker. Similar to [14], we use the weighted focal loss L_{cls} to handle the imbalance between positive and negative training examples for target classification. For

bounding box regression, we use the ℓ_1 and generalized IoU loss functions, denoted by L_{ℓ_1} and L_{giou}, respectively. We define the overall loss function as,

$$L_{total} = L_{cls} + \lambda_1 \cdot L_{\ell_1} + \lambda_2 \cdot L_{giou}, \tag{31}$$

where λ_1 and λ_2 are the hyperparameters controlling the relative impact of L_{ℓ_1} and L_{giou} on the overall training loss.

5.4 Implementation Details and Experimental Results

In this section, we discuss the implementation details of our *MVT* tracker and compares its results with related lightweight trackers. We also present the ablation study results for the proposed feature fusion, along with an attribute-based analysis. Finally, we analyze the *fps* gain obtained by using additional inference frameworks and compare *MVT*'s performance against popular ResNet-based heavy trackers.

5.4.1 Implementation Details

We set the dimensions of the inputs to our *MVT* backbone, i.e., Z_{in} and X_{in} from Section 5.3.1, to 128×128 and 256×256, respectively. We divide our *MVT* backbone into five layers with *layer-ids* for notation convenience, as shown in Figure 5.1. The number of channels in the feature maps increases along these five layers as $\{3 \rightarrow 16, 16 \rightarrow 32, 32 \rightarrow 64, 64 \rightarrow 96, 96 \rightarrow 128\}$. We set the number of transformer blocks for the proposed Siam-MoViT block in *layer*-3 and *layer*-4 to 2 and 4, respectively. We set the parameters $w = h = 2$ for folding and unfolding operations within our Siam-MoViT block. The number of upscaled channels D in the Siam-MoViT block is set to 144 and 192 for *layer*-3 and *layer*-4, respectively. The backbone has a total stride of 16 (i.e., four downsampling operations, each by a factor of two), resulting in feature maps of size 8×8 and 16×16 for the template and search regions, respectively. The *channel-adjust* layer in the neck module, described in Section 5.3.2, upscales the number of channels from 64 to 256.

We use the combined training splits of GOT10k [8], LaSOT [9], TrackingNet [10], and COCO [11] datasets. We apply data augmentation (horizontal flip and scale jittering) to generate training image pairs for the still images in the COCO train dataset. We use *AdamW* [132] as the optimizer with a weight decay of 10^{-4}. We trained our model for 300 epochs with 60000 image pairs per epoch, sampled from the training dataset. We use the validation split of GOT10k to compute the values of L_{cls}, L_{ℓ_1}, and L_{giou} from Eq. 31 during training to examine the possibility of overfitting. We set the initial learning rate lr to 4×10^{-4} and

reduce it by a factor of 10 after 240 epochs. We keep the lr for the backbone module 0.1 times the lr for rest of the network throughout training. We initialize the backbone using the weights of the pre-trained MobileViT model provided by its authors [125]. Same as [125], we do not use positional embeddings for the transformer blocks in our MVT backbone. We set the hyperparameters λ_1 and λ_2 in Eq. 31 to 5 and 2, respectively, as in [14]. We use a single Nvidia Telsa V100 GPU (32GB) for training and set the batch size to 128.

Our choice of optimizer and hyperparameters is based on the training settings typically used by the related trackers. We set our batch size based on the maximum number of images that can be loaded onto the GPU used for training the model. We experimented using the Ray-Tuner package in Pytorch [133] to search for the best set of hyperparameters jointly. First, the hyperparameter search was time-consuming due to the sheer volume. Second, due to a strong inter-dependency between some of the hyperparameters (*e.g.*, batch size and learning rate), it was challenging to find the optimal set using randomized search.

During inference, we define the search space at frame t by extracting an image patch around the estimated target location at frame $t-1$, four times the area of the target template. We apply a Hanning window on the classification score map $\mathcal{R}$ as the post-processing step. After this multiplication, we determine the location of the highest value in $\mathcal{R}$ as the target location, and we choose the corresponding bounding box as the tracker output. We define the target annotation from the first frame as the template and do not perform any model update. We generate the CPU-based results using a 12th Gen Intel(R) Core i9 processor.

5.4.2 Comparison to Related Work

To demonstrate the effectiveness of the proposed MVT, we evaluate its performance across multiple datasets, including GOT10k-test [8], TrackingNet-test [10], LaSOT-test [9], OTB100 [6], and TC128 [93]. As explained in Section 1.3, the test videos in GOT10k have non-overlapping target classes compared to their training videos, to promote generalization during tracker development. Also, GOT10k and TrackingNet datasets sequester the test set annotations and provide an online evaluation server to submit the tracker results to ensure a fair evaluation. For the LaSOT, OTB100, and TC128 datasets, we compute the evaluation metrics by using publicly available groundtruth labels. While quantifying the tracker performance on these datasets, we adhere to the evaluation metrics recommended by their respective authors, as outlined in Table 1.1

We compare the results of the proposed MVT with the related lightweight trackers: LightTrack [121], Stark-Lightning [13], FEAR-XS [122], and E.T.Track [127], evaluated using the pretrained models provided by their authors. The results, as summarized in Table 5.1,

Tracker	Source	GOT10k (s)			TrackingNet (s)			LaSOT			OTB100	TC128	*fps*
		AO	$SR_{0.50}$	$SR_{0.75}$	*AUC*	P_{norm}	*P*	*AUC*	P_{norm}	*P*	*AUC*	*AUC*	(CPU)
LightTrack	CVPR'21	0.582	0.668	0.442	0.729	0.793	0.699	0.522	0.583	0.517	0.659	0.558	42.2
FEAR-XS	ECCV'22	0.573	0.681	0.455	0.715	0.805	0.699	0.501	0.594	0.523	0.664	0.591	42.3
Stark-Lightning	ICCV'21	0.596	0.696	0.479	0.727	0.779	0.674	0.578	0.660	0.574	0.628	0.573	49.9
E.T.Track	WACV'23	0.566	0.646	0.425	0.740	0.798	0.698	**0.589**	**0.670**	**0.603**	0.671	0.569	44.6
MVT (ours)	-	**0.627**	**0.723**	**0.577**	**0.756**	**0.816**	**0.720**	0.561	0.634	0.572	**0.675**	**0.599**	29.5

Table 5.1: Comparison of related lightweight SN trackers with our *MVT* on server-based GOT10k-test and TrackingNet-test, LaSOT-test, OTB100, and TC128 datasets. The best and second-best results are highlighted in **red** and blue, respectively.

showcase the superior performance of *MVT* across various metrics and datasets. Notably, *MVT* outperforms all other lightweight trackers on the server-based test set of GOT10k and TrackingNet, demonstrating its generalization capability. On **GOT10k-test**, specifically designed with non-overlapping target classes compared to the training set, *MVT* outperforms the second-best tracker by margins of 3.1%, 2.7%, and 9.8% in terms of *OR*, $SR_{0.50}$, and $SR_{0.75}$, respectively. This indicates *MVT*'s superior generalization to novel object classes, emphasizing the impact of feature fusion in its backbone compared to other two-stream lightweight trackers.

On the **TrackingNet-test** dataset, *MVT* exhibits a margin of approximately 2% in AUC, *P*, and P_{norm} over its closest competitor, LightTrack. In the **LaSOT-test** dataset, *MVT* outperforms LightTrack by 3.9% in *AUC* and surpasses FEAR-XS with a 6% margin. However, it is worth noting that *MVT* falls short of transformer-based related trackers on this dataset, indicating potential areas for improvement, especially while tracking objects in long videos. On short-term tracking benchmarks, i.e., **OTB100** and **TC128**, *MVT* achieves superior results compared to all related trackers, and showcases a higher *AUC* score of 1.9% and 2.6%, respectively, on average.

Across diverse datasets and performance metrics, *MVT* achieves top scores in the majority of cases (8 out of 11). E.T.Track, its closest competitor, secures the best score in 3/11 instances and second-best in 2 out of 11 cases. While *MVT* achieves near real-time inference speed at 29.5 *fps* on a CPU, it lags behind the related trackers due to the necessity of evaluating template features at every frame, impacting overall tracking speed.

5.4.3 Ablation Study Related to the Proposed Feature Fusion

In this section, we analyze the effectiveness of the proposed feature fusion technique in our *MVT* backbone through an ablation study, where we evaluate the performance of our tracker without fusing the template and search region features inside the proposed Siam-MoViT block (*cf.* Figure 5.1). The ablation results on large-scale datasets, including GOT10k, TrackingNet, and LaSOT, are summarized in Table 5.2. The table demonstrates that the

feature fusion in backbone	GOT10k		TrackingNet			LaSOT		
	$AO \uparrow$	$SR_{0.50} \uparrow$	$AUC \uparrow$	$P_{norm} \uparrow$	$P \uparrow$	$AUC \uparrow$	$P_{norm} \uparrow$	$P \uparrow$
✗	0.600	0.703	0.749	0.800	0.702	0.539	0.610	0.542
✓(ours)	**0.627**	**0.742**	**0.756**	**0.816**	**0.720**	**0.561**	**0.634**	**0.572**

Table 5.2: Ablation study results related to the proposed feature fusion in our *MVT* backbone. Best results are highlighted in red.

proposed feature fusion significantly enhances the *AO* (or the equivalent metric *AUC*) by 2.2% on average across all datasets. This improvement showcases the effectiveness of the proposed feature fusion by applying self-attention on concatenated features in our *MVT* backbone. Proposed feature fusion enables global relational modeling *within* and *between* the template and search regions, leading to the generation of superior features for accurate target localization. The ablation study results suggest that the proposed feature fusion mechanism contributes substantially to the overall tracking performance of *MVT*.

5.4.4 Attribute Analysis

To analyze the performance of the proposed *MVT* tracker against various video challenges (or attributes), we compute its Failure Rate (*FR*) for attributes annotated under the LaSOT dataset, namely Aspect Ratio Change (*ARC*), Background Clutter (*BC*), Camera Motion (*CM*), Deformation (*DEF*), Fast Motion (*FM*), Full Occlusion (*FOC*), Illumination Variation (*IV*), Low Resolution (*LR*), Motion Blur (*MB*), Out-of-View (*OV*), Partial Occlusion (*POC*), Rotation (*ROT*), Scale Variation (*SV*), and Viewpoint Change (*VC*). From Figure 5.2, we can see that our *MVT* is most resilient to target deformation (*DEF*) and appearance changes (*VC*). It has a high *FR* on attribute *FM* since we use a Hanning window on the classification score map during target localization. However, not using the Hanning window deteriorates the performance of our tracker against *BC* and increases the overall *FR*, as we observed from our experiments. Also, our *MVT* has a higher *FR* for videos under the attribute *LR*. These videos contain small, texture-less target objects such as *volleyball* and *yo-yo*, which are generally fast-moving (i.e., *FM*) and are sensitive to *BC*. State-of-the-art trackers address the challenges of *FM*, *LR*, and *BC* with deep features and by defining a larger search area to avoid target loss, but these improvements come at the expense of higher model complexity and memory footprint.

5.4.5 Tracker Inference Using Different Frameworks

To further optimize the speed of our tracker during CPU and GPU-based inference, we evaluate our *MVT* using two additional frameworks: ONNX-Runtime [18] and TensorRT [96],

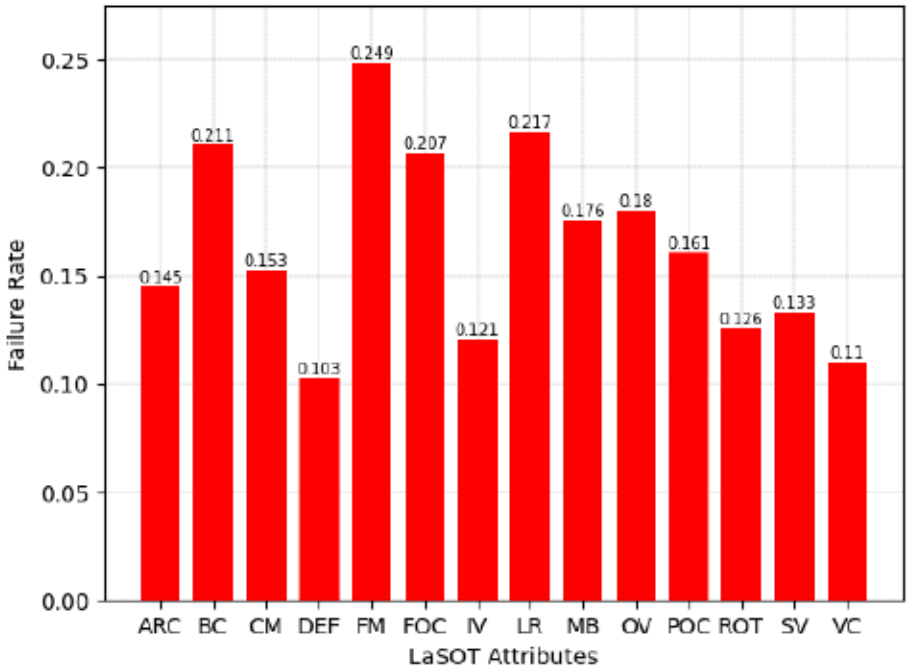

Figure 5.2: Analyzing the performance of our *MVT*, based on its Failure Rate (FR) for different attributes on the LaSOT test dataset. The average FR is 0.137.

apart from the native PyTorch [133], which we used for tracker implementation and training. ONNX-Runtime is a cross-platform inference accelerator that optimizes speed by leveraging the computational capabilities of available hardware, especially CPUs. TensorRT is a similar framework by Nvidia, designed to optimize model performance on Nvidia GPUs. These frameworks perform operations such as precision calibration, kernel auto-tuning, weight quantization, layer and tensor fusion [134], to reduce model latency and memory footprint during inference with minimal or no loss in accuracy.

Table 5.3 quantifies the impact of these frameworks on model inference speed relative to the native PyTorch [133]. For our *MVT* tracker, the use of ONNX-Runtime and TensorRT frameworks improved the *fps* by 2.3× on CPU and 1.7× on GPU, respectively, with less than a 1% loss in accuracy. Overall, leveraging ONNX-Runtime and TensorRT significantly enhance tracker *fps* compared to the PyTorch baseline, demonstrating substantial inference speed improvements with minimal impact on accuracy.

5.4.6 Comparison to ResNet50-based Trackers

To highlight the effectiveness of the proposed lightweight tracker further, we conduct a comparative analysis against DiMP-50 [27] and Ocean [68] trackers, both known for their superior performance in VOT and utilizing ResNet50 as their backbone. The evaluation is carried out

Tracker	Hardware	Framework	GOT10k-test		*fps*	Improvement
			$AO \uparrow$	$SR_{0.50} \uparrow$		w.r.t. baseline
MVT	CPU	Pytorch	0.627	0.723	29.5	-
		ONNX-Runtime	0.624	0.718	68.6	2.3×
	GPU	Pytorch	0.627	0.723	174.4	-
		TensorRT	0.617	0.724	302.2	1.7×

Table 5.3: Comparing the improvement in tracker inference speed (in *fps*) using ONNX-Runtime and TensorRT frameworks, relative to the native Pytorch as baseline. We also present the performance metrics for the GOT10k-test using these frameworks.

Tracker	GOT10k		TrackingNet		#params ↓	*fps*	
	$AO \uparrow$	$SR_{0.50} \uparrow$	$AUC \uparrow$	$P_{norm} \uparrow$	(in millions)	GPU ↑	CPU ↑
DiMP-50 [27]	0.611	0.717	0.740	0.801	26.1	61.5	15.0
Ocean [68]	0.615	**0.735**	0.693	0.794	44.3	127.8	10.1
MVT (ours)	**0.627**	0.723	**0.756**	**0.816**	**5.5**	**174.4**	**29.5**

Table 5.4: Comparison of our *MVT* with the ResNet50-based trackers on GOT10k and TrackingNet test datasets. Best results in accuracy and complexity (i.e., # of parameters and *fps*) are highlighted in red.

using server-based GOT10k and TrackingNet test datasets, providing a comprehensive assessment across different tracking scenarios. In Table 5.4, we present the comparative results, highlighting key performance metrics associated with GOT10k and TrackingNet datasets. Additionally, we provide insights into the model complexity by reporting the number of parameters and the corresponding *fps* values achieved during inference on both CPU and GPU platforms.

Notably, our proposed *MVT* outperforms both DiMP-50 and Ocean in terms of *AO* and the equivalent *AUC* metric on both datasets. Despite being equipped with a significantly leaner architecture with only 5.5 million parameters compared to DiMP-50 (26.1 million) and Ocean (44.3 million), *MVT* demonstrates superior tracking accuracy. Furthermore, our *MVT* achieves near real-time performance on a CPU, surpassing the speeds of DiMP-50 and Ocean by at least 2 times. *MVT* also leverages the computational capabilities of a GPU to achieve an impressive speed of 174.4 fps, outpacing both baseline trackers significantly. These results underscore the impact of our novel hybrid CNN and transformer-based feature extraction block in the backbone, while maintaining computational efficiency for high-speed tracking on both CPU and GPU platforms.

5.5 Summary

In this chapter, we introduced a novel VOT algorithm, named *MVT*, that uses Mobile Vision Transformers as the backbone. We also proposed the Siam-MoViT block to fuse the template and search regions in the tracker backbone, thereby enhancing the quality of feature encodings for target localization. Our simulation results showed that the proposed tracker performed better than the related lightweight trackers on the large-scale GOT10k and TrackingNet datasets, as well as the short-term OTB100 and TC128 benchmarks, showcasing the effectiveness of the proposed tracking method. Despite having 4.7× fewer model parameters, our *MVT* performs better than the popular DiMP-50 tracker, while running at least 2× its speed during CPU and GPU-based inference. Ablation studies further highlighted the significance of the proposed feature fusion on our tracker's performance.

To discuss the limitations, *MVT* exhibits suboptimal performance on the long-term tracking dataset, LaSOT. This can be attributed to the CNN-based decoder (or head module) used for target localization, which ineffectively capture non-local associations crucial for localization under scenarios involving drastic target shape variations and heavy occlusion. Such failures early in a long video can lead to poor performance throughout the rest of the sequence. Another limitation is the computational complexity of dense matrix-to-matrix multiplication operations during the self-attention computation (see Eq. 5), which lowers the inference speed of the proposed tracker. The next Chapter 6 addresses these limitations.

Chapter 6

Separable Self and Mixed Attention Transformer-based Tracking

6.1 Introduction

In this chapter (based on our paper [98]), we address the limitations encountered with our *MVT* tracker from Chapter 5 and introduce an enhanced high-speed tracker termed *SMAT*. While the deployment of cascaded convolutional and transformer blocks in the *MVT* backbone effectively modeled spatially local and long-range dependencies, the purely Convolutional Neural Network-based (CNN) predictor head struggled to capture non-local associations within the encoded features, leading to suboptimal performance. Although transformers can be a remedy for this issue, the computational complexity of standard transformers posed a challenge, causing our tracker model's latency to drop below 30 *fps* on a CPU.

To tackle the computational complexity associated with standard transformers, various alternative solutions have emerged. Many of these solutions aim to approximate the standard self-attention computation using a set of efficient operations with lower latency. However, the extent of improvement in *fps* achieved by these approximation algorithms depends on the hardware used and the number of tokens involved during attention computation. In this regard, Mehta et al. [135] conducted a comparative analysis of Linformer [136], a popular mechanism that approximated self-attention computation with linear complexity. The study found that during CPU-based inference, Linformer exhibited worse latency than standard transformers for #tokens $= 256$, which is comparable to the number of tokens in our *MVT*. To optimize the efficiency of attention computation in the context of vision transformers, [135] proposed an efficient vision transformer block using separable elementwise operation-based

attention computation, which demonstrated a 37% faster performance than standard transformers on a CPU. Leveraging these advancements, we enhance the performance of our *MVT* tracker by replacing its CNN-based head module with an efficient separable self-attention transformer-based head. Additionally, to further boost the computational efficiency, we replace the standard transformers in *MVT*'s backbone with those deploying separable mixed-attention. With these contributions, we introduce *SMAT*, a lightweight real-time tracking algorithm that concurrently deploys a hybrid of CNN and transformers in its backbone and head modules for the first time.

6.2 Related Work

Several SN lightweight trackers have been proposed in recent years, aiming to achieve real-time tracking on CPUs and edge devices [13, 121, 122, 127, 137, 138]. Notably, fewer lightweight SN trackers use transformer modules in their backbone or head architecture. MixFormerV2 [138] presented a fully transformer-based backbone for efficient tracking, relying on knowledge-distillation [139] and progressive model-depth pruning. E.T.Track [127] deployed a novel Exemplar Transformer in its predictor head for real-time tracking, utilizing a single instance-level attention layer in the transformer block.

In comparison to other related trackers, MixFormerV2 and E.T.Track closely align with our work. However, our approach differentiates itself by utilizing a cascade of CNNs and efficient Vision Transformer blocks (ViT) [135] with separable attention in the backbone, deviating from the fully transformer-based backbone employed by MixFormerV2 [138]. As our experimental results will demonstrate, the fusion of CNN and transformers in our approach yields superior tracking performance. Moreover, the iterative nature of knowledge-distillation and model pruning by MixFormerV2 requires multiple rounds of model training, whereas our tracker model needs to be trained only once. Compared to the Exemplar Transformer head module by E.T.Track [127], our separable self-attention predictor head is 3× compact in terms of parameters (5.7 Million for E.T.Track versus 1.8 Million for our *SMAT*). Notably, E.T.Track employs a post-processing step during the inference phase to refine predicted bounding boxes, involving penalization of substantial changes in target size and aspect ratio between consecutive frames, which necessitates fine-tuning of related hyperparameters. In contrast, our post-processing requires no fine-tuning as we apply a Hanning window on the classification score map $\mathcal{R}$ (*cf.* Figure 6.4) to penalize significantly large target displacement predictions.

To summarize our contributions, we are the first to propose:

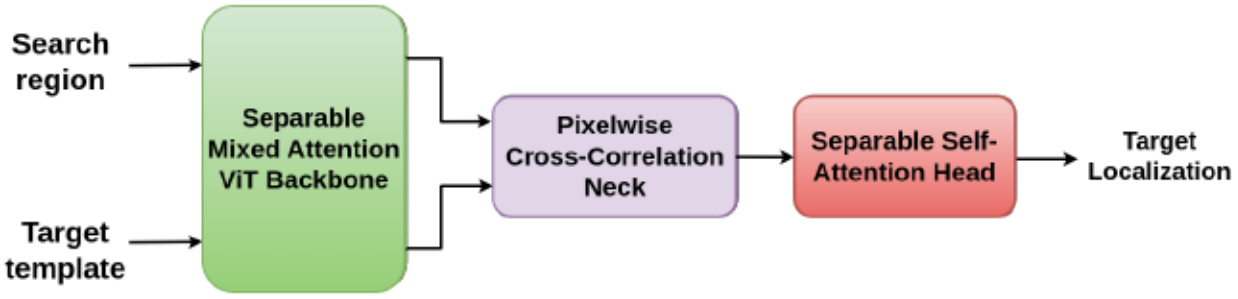

Figure 6.1: Block diagram of the proposed *SMAT*. The separable mixed attention transformer-based backbone jointly performs feature extraction and fusion of template and search regions. The separable transformer-based head models long-range dependencies within the fused encoding to generate target localization results.

- A separable mixed attention ViT-based backbone for joint feature extraction and information fusion between the template and search regions. Our approach combines the strengths of CNNs and ViTs, yielding superior feature encoding for accurate tracking without bloating the backbone latency.
- A separable self-attention ViT-based predictor head, capturing the global dependencies within the fused feature encoding for accurate bounding-box prediction.
- A concurrent deployment of CNN and transformer-based hybrid module in the backbone and predictor head for lightweight tracking.

6.3 Proposed Tracker *SMAT*

6.3.1 Overview

The pipeline of the proposed *SMAT* tracker is shown in Figure 6.1. We employ a cascaded arrangement of CNNs and ViT blocks with separable attention [135] in the proposed tracker backbone. Such a hybrid design combines the merits of convolutions (i.e., learning the spatially-local representations) and transformers (i.e., modeling the long-range dependencies) with fewer parameters compared to the fully transformer-based backbone architecture, such as [14, 15]. Apart from generating the feature representation, proposed backbone facilitates the exchange of information between the target template and search region by computing mixed attention [15] in the ViT block without bloating the backbone latency. Our predictor head efficiently performs global contextual modeling of fused feature encoding using separable self-attention units and improves the localization accuracy compared to the fully convolutional methods, as shown in [127]. With these contributions, we propose

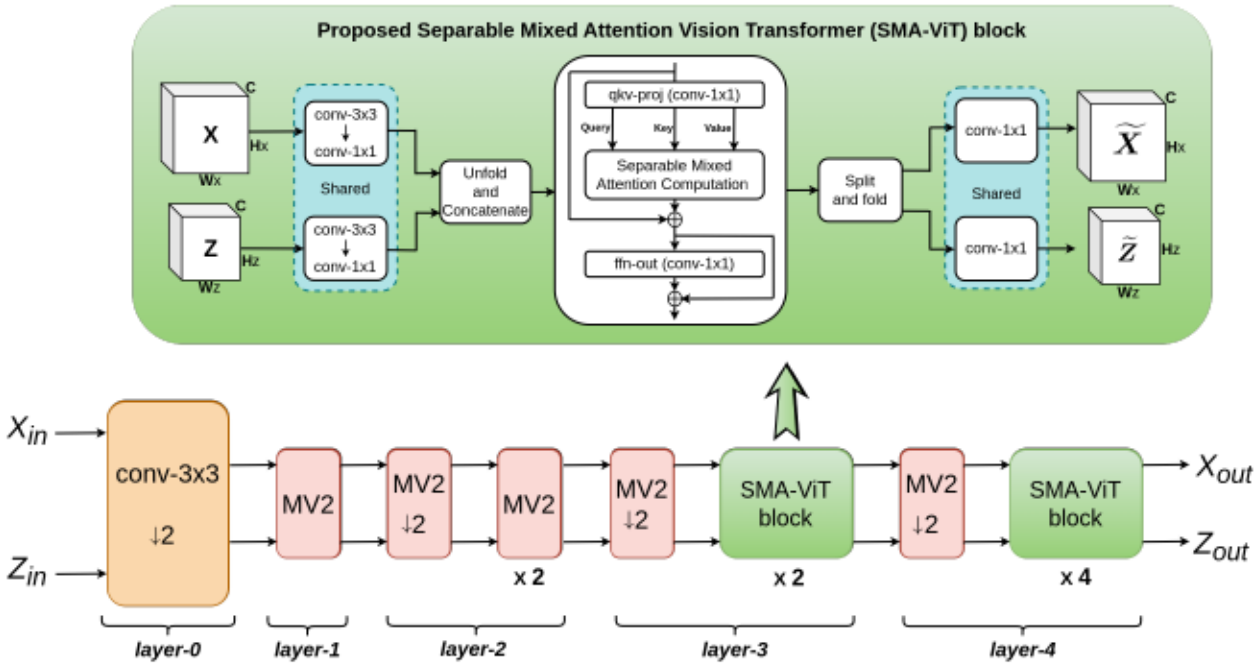

Figure 6.2: The proposed backbone of our tracker and its Separable Mixed Attention Vision Transformer (SMA-ViT) block. MV2 indicates the CNN-based MobileNetV2 block and ↓2 denotes spatial downsampling by a factor of 2. Within the SMA-ViT block, *qkv-proj* denotes the set of three 1×1 convolutional filters to generate the *Query*, *Key*, and *Value* for attention computation. The mixed attention output is passed through a 1×1 convolutional *ffn-out* block to generate the output of the transformer layer.

a lightweight self and mixed attention transformers-based tracker, *SMAT*, running beyond real-time speed on a CPU.

6.3.2 Proposed Backbone

This section delves into the backbone architecture of the proposed *SMAT* tracker, illustrated in Figure 6.2. The *SMAT* backbone receives two images as its input; one is the target template Z_{in}, and the other is the search region for target localization, X_{in}. We then apply CNN-based Inverted Residual or MobileNetV2 blocks [7] (referred to as MV2 in Figure 6.2) on both Z_{in} and X_{in}. These MV2 blocks not only efficiently generate feature representations capturing spatially local relations but also reduce the spatial dimensionality of the input images via pooling operations, thereby generating low-dimensional feature representations conducive to our Separable Mixed Attention-based Vision Transformer (SMA-ViT) block. The sequential arrangement of MV2 and SMA-ViT blocks in the proposed backbone is the same as MobileViTv2 [135] pipeline.

Let $Z \in R^{W_z \times H_z \times C}$ and $X \in R^{W_x \times H_x \times C}$ be inputs to the SMA-ViT block, from the

template and search regions, respectively. Inside the proposed block, we initially pass Z and X through a series of 3×3 and 1×1 CNN layers with shared weights to project the number of channels in Z and X from C to d. The output of these CNN blocks is tokenized and concatenated to form k tokens, facilitating mixed attention between template and search regions. Inside the transformer layer, we first apply a set of three 1×1 convolutional filters (denoted as *qkv-proj* in Figure 6.2) to generate the query $\mathcal{Q} \in R^{k \times 1}$, the key $\mathcal{K} \in R^{k \times d}$, and the value $\mathcal{V} \in R^{k \times d}$. Then, we apply the softmax operation on the query vector $\mathcal{Q}$ and broadcast along its column (i.e., the element in i^{th} row is repeated d times along the column dimension) to generate $\tilde{\mathcal{Q}} \in R^{k \times d}$. Using $\tilde{\mathcal{Q}}$ and $\mathcal{K}$, the context vector $\mathcal{A} \in R^{1 \times d}$ is computed as

$$\mathcal{A} = \sum_{k} \tilde{\mathcal{Q}} \odot \mathcal{K}, \tag{32}$$

where $\odot$ denotes the element-wise multiplication and $\sum_k$ indicates summation across the rows. The context vector $\mathcal{A}$ is broadcasted along its rows to create $\tilde{\mathcal{A}} \in R^{k \times d}$, which is used to compute the mixed attention $\mathcal{M} \in R^{k \times d}$ as

$$\mathcal{M} = \tilde{\mathcal{A}} \odot \mathrm{ReLU}(\mathcal{V}). \tag{33}$$

The elementwise multiplication operations in Eq. 32 and Eq. 33 reduce the latency of the separable transformer layer, shown in Figure 6.2, when compared to the dense matrix-to-matrix multiplication-based attention computation in Eq. 5. Also, computing the mixed attention on the concatenated features concurrently models the global interactions *within* (i.e., self) and *between* (i.e., cross) the target template and the search area. Therefore, mixed attention requires fewer transformer block computations than separately computing the self and cross-attention. Similar to [2], we employ a residual connection [35] around the attention computation block. We pass the output of the residual connection through a 1×1 convolutional feedforward network (denoted as *ffn-out* in Figure 6.2) to generate the output of the separable transformer layer. We split the resulting feature map to separate the template and search region features with d channels. Finally, we re-project the number of channels from d to C by applying a shared 1×1 convolutional filter on the separated feature maps to generate $\tilde{X}$ and $\tilde{Z}$ as the output. With four downsampling operations, denoted as $\downarrow 2$ in Figure 6.2, the spatial dimension of the features generated by our backbone, i.e., Z_{out} and X_{out} is 8×8 and 16×16, respectively.

To illustrate the effectiveness of the proposed backbone, we visualize the attention maps (i.e., $\mathcal{M}$ from Eq. 33) alongside corresponding tracker output for specific sequences from the LaSOT test dataset, as depicted in Figure 6.3. The images on the left contain the target

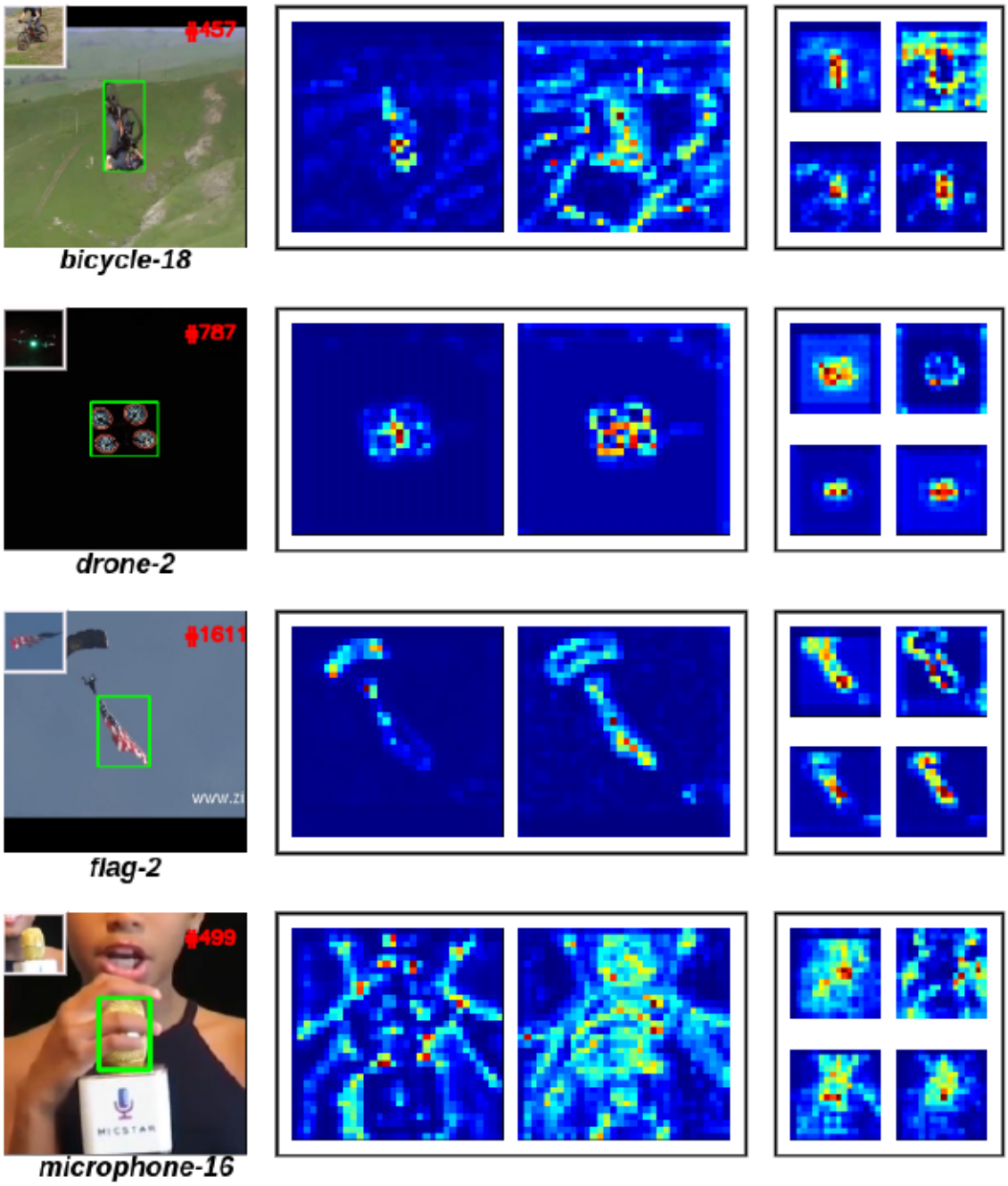

Figure 6.3: Visualizing the bounding box output (left) and the corresponding attention maps (center and right) for the proposed *SMAT* tracker. The larger values in the attention map are denoted by red color, while the smaller values are represented by blue color.

template (top-left corner), the search region at frame #t, and the tracker output. The images in the center and right indicate the attention maps corresponding to the SMA-ViT blocks at *layer-3* (two transformers) and *layer-4* (four transformers) of the backbone, respectively (*cf.* Figure 6.2). For the examples shown in Figure 6.3, the target object is impacted by a video

challenge, i.e., Aspect Ratio Change for *bicycle-18*, Illumination Variation for *drone-2*, Target Deformation for *flag-2*, and Partial Occlusion for *microphone-16*. Despite the influence of these attributes, our *SMAT* successfully locates the target object. For the example shown in the last row of Figure 6.3, the target is partially occluded by an external object. In this case, the tracker focuses on the visual cues around the target object, as seen from the attention maps in the center. This information is processed by the subsequent transformer blocks of in the backbone to produce stronger attention values in the target center and generate an accurate bounding box.

6.3.3 Tracker's Neck

The computation of mixed attention in the SMA-ViT block facilitates implicit relation modeling (or feature fusion) during feature extraction, hence it avoids needing a parameter-heavy neck module for the subsequent relation modeling between the template and search region features. For a parameter-efficient fusion of features X_{out} and Z_{out} generated by the proposed backbone, we use the *parameter-free* pixel-wise cross-correlation [63] operation. As in LightTrack [121], we compute the fused feature encoding F as,

$$F = PWCorr(X_{out}, Z_{out}). \tag{34}$$

where $PWCorr$ denotes the pixel-wise cross-correlation operation. We apply a 1×1 convolution filter on the fused encoding to transform the number of channels to C_h. To perform target localization, we pass the resulting encoding F to the proposed predictor head in Section 6.3.4.

6.3.4 Proposed Predictor Head

The architecture of the proposed predictor head is shown in Figure 6.4. It has two branches for target classification and bounding-box regression. In contrast to a fully CNN-based approach, we design these branches using a cascade of convolutional and transformer layers. The choice of CNN layers is motivated by their effectiveness in capturing local relationships within the fused feature representation. However, CNNs have limitations in modeling non-local associations effectively. On the other hand, transformer layers explicitly model long-range global interactions within features, making them beneficial for scenarios involving significant target shape variations and heavy occlusion [13]. The cascade of convolutional and transformer layers combines the strengths of both modeling approaches to produce high-quality tracking results, especially in challenging videos.

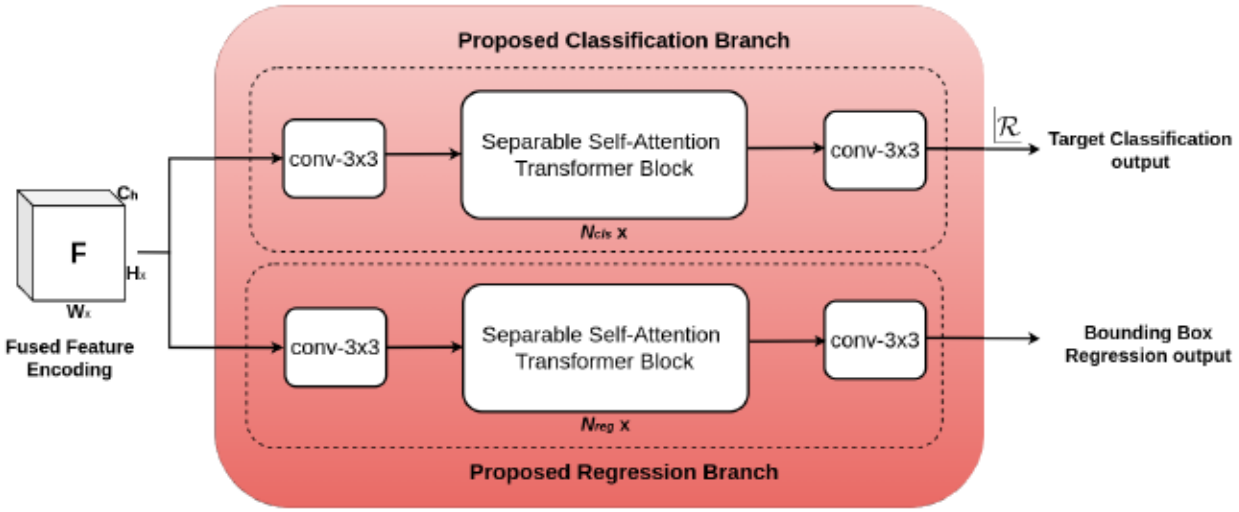

Figure 6.4: Proposed Separable Self-Attention Transformer-based Predictor Head. It receives the fused feature encoding, i.e., output of the neck module, as its input and utilizes two branches for target classification and bounding-box regression. N_{cls} and N_{reg} refer to the number of transformer blocks in classification and regression branches, respectively. $\mathcal{R}$ indicates the two-dimensional score map generated by the target classification branch.

The predictor head takes the fused feature encoding F in Eq. 34 (i.e., output of the neck module) as its input. As the first step, we apply a 3×3 convolutional filter for the proposed predictor head on the fused encoding F from Eq. 34 to extract spatially local feature representations. We then tokenize the filter output and pass it through a stack of N_{cls} and N_{reg} separable self-attention transformer layers in classification and regression branches, respectively. For these transformer blocks, computation of attention is similar to the transformer layer in the SMA-ViT block from Section 6.3.2. However, in this case, self-attention is calculated to model long-range dependencies within the fused feature encoding F for accurate target state estimation. The output of the self-attention transformer layer is passed through a 3×3 CNN layer to generate a score map $\mathcal{R}$ for the classification branch, and the local offset and the normalized bounding-box size for the regression branch, following the approach in [14]. The combination of local and long-range contextual modeling by our predictor head improves tracking performance without significantly increasing the overall model latency.

For the proposed head module, we set the number of channels in the fused feature encoding, i.e., C_h in Figure 6.4, to 128. We set the number of transformer layers N_{cls} and N_{reg} in the classification and regression branches to 2 and 4, respectively. The reason for defining N_{reg} twice the value of N_{cls} is because the regression head predicts twice the variables, i.e., local offset and bounding-box size, compared to the classification branch predicting the target center.

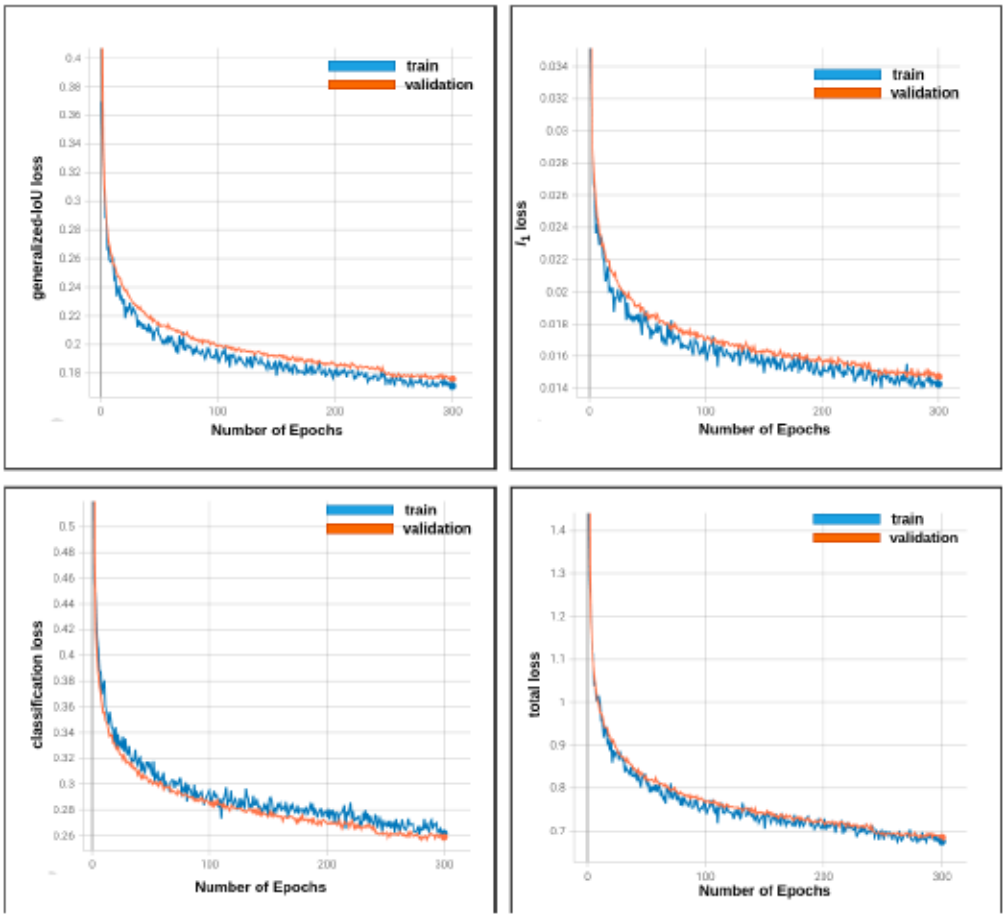

Figure 6.5: Visualizing the generalized IoU L_{giou} (top-left), L_{ℓ_1} (top-right), and classification L_{cls} (bottom-left) loss functions versus the number of epochs for the combined training splits of GOT10k [8], LaSOT [9], TrackingNet [10], and COCO [11] datasets. We use the validation split of GOT10k dataset to compute the validation loss values. The bottom-right plot illustrates the total loss L_{total}, calculated as the weighted sum of these loss functions, as described in Eq. 31. It is seen that the training and validation loss values decrease for a certain number of epochs until there is no significant change, indicating no overfitting.

6.3.5 Tracker Training

During training of our *SMAT* tracker, we use the loss function from Eq. 31 on the classification and regression output by the head module. Based on the discrepancies between the predicted output and groundtruth annotations measured by the loss function, we update the model parameters using the backpropagation algorithm. To monitor the potential occurrence of overfitting, we evaluate the values of loss functions on the GOT10k validation videos, as indicated in Figure 6.5.

We train the proposed *SMAT* on the combined training splits of GOT10k [8], LaSOT [9], TrackingNet [10], and COCO [11] datasets. To diversify the training data, we apply data

augmentation techniques such as horizontal flipping and scale jittering, particularly for generating training image pairs from still images in the COCO train dataset. We conduct model training on a single Nvidia Tesla V100 GPU (32GB). We use the following hyperparameters during training.

- We train our model for 300 epochs, with 60000 image pairs randomly sampled from the training dataset during each epoch.
- We set the initial learning rate lr to 4×10^{-4} and keep the lr for the backbone module 0.1 times the lr for the rest of the network throughout training.
- We utilize *Adam-W* [132] as the optimizer with a weight decay of 10^{-4}.
- We use Pytorch's *StepLR* scheduler to reduce the lr by a factor of 10 after 240 epochs.
- We initialize the backbone of our *SMAT* using the pre-trained MobileViTv2 [135] model weights, provided by their authors.
- We set the batch size set to 128.

Note that we do not finetune these hyperparameters specifically for *SMAT*, adhering to the settings established for our *MVT* tracker in Chapter 5.

6.3.6 Tracker Inference

During the inference phase, we adopt the annotation from the first frame in the video as the target template, and no model update is performed during the tracking process. To define the search region at frame t, we extract a region around tracker output at frame $t-1$, extending four times the target size. The extracted region is resized to 256×256 and utilized as the search image at frame t. As a post-processing step, we apply a Hanning window on the classification score map $\mathcal{R}$ (*cf.* Figure 6.4). This windowing operation is instrumental in penalizing significantly large target displacement predictions, contributing to enhanced tracking performance.

6.4 Experimental Results and Analysis

6.4.1 Experimental Setup

To assess the performance of the proposed *SMAT*, we evaluate our tracker on the test-split of GOT10k [8], LaSOT [9], TrackingNet [10], NfS30 [90], UAV123 [88], AVisT [89], OTB100 [6],

and TC128 [93] datasets. We use the metrics recommended by the corresponding dataset authors to quantify the tracker performance during our evaluation (*cf.* Table 1.1). To ensure fair tracker evaluation and avoid finetuning of parameters on the test data, GOT10k and TrackingNet sequester the ground-truth annotations for their test videos. Therefore, we generate the metrics for these datasets by submitting the raw tracker results to the remote evaluation server. The groundtruth annotations are publicly available for the LaSOT-test, NfS30, UAV123, AVisT, OTB100, and TC128 datasets. The development of the tracker code is carried out using PyTorch [133]. Additionally, we implement our tracker inference using ONNX-Runtime [18] to demonstrate higher speed achieved. We use a 12th Gen Intel(R) Core i9 processor to generate the CPU-based inference results and a single Nvidia RTX 3090 for GPU-based results.

6.4.2 Comparison to Related Lightweight Trackers

We compare the results of the proposed *SMAT* against the related lightweight trackers: LightTrack [121], Stark-Lightning [13], FEAR-XS [122], HCAT [137], E.T.Track [127], and MixFormerV2-S [138], evaluated using the pretrained models provided by their authors. Note that MixFormerV2-S [138] is a lightweight version derived from the heavyweight MixFormer-L [15]. For the sake of completeness, we also add the performance metric values of our *MVT* from Chapter 5. From Table 6.1, we can see that the proposed *SMAT* outperforms the related trackers on all eight test datasets: GOT10k-test, TrackingNet-test, LaSOT-test, NfS30, UAV123, AVisT, OTB100, and TC128. Notably, no related tracker demonstrates consistently superior performance across the eight datasets. HCAT exhibits the second-best results in 8 out of 18 metrics across all datasets, while MixFormerV2-S scores the second-best in 6 out of 18 cases.

The qualitative results on various datasets highlight the performance of our *SMAT* across diverse set of tracking-related challenges in videos:

- **GOT10k-test:** Since the dataset has videos with target object categories unseen during training, it is well-suited for evaluating tracker generalization. On this dataset, our *SMAT* showcases an average improvement of 3.5% in AO, 4% in $SR_{0.50}$, and 5.8% in $SR_{0.75}$ than MixFormerV2-S and HCAT. MixFormerV2-S and E.T.Track are closely related to our *SMAT* since they deploy a fully transformer-based backbone and predictor head, respectively. By concurrently deploying a CNN and transformer-based backbone and head module, our *SMAT* outperforms both E.T.Track and MixFormerV2 on average by a significant margin of 6.8% in AO, 8.8% in $SR_{0.50}$, and 12.4% in $SR_{0.75}$.

- **TrackingNet-test:** This benchmark consists of 511 challenging videos curated from

Tracker	Source	Transformer based	GOT10k (s)			TrackingNet (s)			LaSOT			*fps* (CPU)
			AO	$SR_{0.50}$	$SR_{0.75}$	*AUC*	P_{norm}	*P*	*AUC*	P_{norm}	*P*	
LightTrack	CVPR'21	✗	0.582	0.668	0.442	0.729	0.793	0.699	0.522	0.583	0.517	42.2
FEAR-XS	ECCV'22	✗	0.573	0.681	0.455	0.715	0.805	0.699	0.501	0.594	0.523	42.3
Stark-Lightning	ICCV'21	neck	0.596	0.696	0.479	0.727	0.779	0.674	0.578	0.660	0.574	49.9
HCAT	ECCV'22	neck	**0.634**	**0.743**	0.558	0.763	**0.824**	**0.726**	0.590	0.683	0.605	45.0
E.T.Track	WACV'23	head	0.566	0.646	0.425	0.740	0.798	0.698	0.589	0.670	0.603	44.6
MixFormerV2-S	NeurIPS'23	backbone	0.587	0.672	0.482	**0.767**	0.812	0.714	**0.610**	**0.694**	**0.614**	30.0
MVT (ours)	BMVC'23	backbone	0.627	0.723	**0.577**	0.756	0.816	0.720	0.561	0.634	0.572	29.5
SMAT (ours)	-	backbone and head	**0.645**	**0.747**	**0.578**	**0.786**	**0.842**	**0.756**	**0.617**	**0.711**	**0.646**	36.9 (**74.1***)

Tracker	Source	Transformer based	NfS30		UAV123		AVisT			OTB100	TC128	*fps* (CPU)
			AUC	*P*	*AUC*	*P*	*AUC*	OP50	OP75	*AUC*	*AUC*	
LightTrack	CVPR'21	✗	0.565	0.692	0.617	0.799	0.404	0.437	0.242	0.659	0.558	42.2
FEAR-XS	ECCV'22	✗	0.486	0.563	0.610	0.816	0.370	0.421	0.220	0.664	0.591	42.3
Stark-Lightning	ICCV'21	neck	0.596	0.710	0.620	0.820	0.394	0.431	0.223	0.628	0.573	49.9
HCAT	ECCV'22	neck	**0.619**	**0.741**	0.620	0.805	**0.418**	**0.481**	0.263	0.673	0.580	45.0
E.T.Track	WACV'23	head	0.570	0.694	0.626	0.808	0.390	0.412	0.227	0.671	0.569	44.6
MixFormerV2-S	NeurIPS'23	backbone	0.610	0.722	**0.634**	**0.837**	0.396	0.425	0.227	0.622	0.592	30.0
MVT (ours)	BMVC'23	backbone	0.567	0.676	0.608	0.790	0.397	0.434	**0.287**	**0.675**	**0.599**	29.5
SMAT (ours)	-	backbone and head	**0.620**	**0.746**	**0.643**	**0.839**	**0.447**	**0.507**	**0.313**	**0.689**	**0.597**	36.9 (**74.1***)

Table 6.1: Comparison of proposed *SMAT* with related lightweight SN trackers on GOT10k-test (server), TrackingNet-test (server), LaSOT-test, NfS30, UAV123, AVisT, OTB100, and TC128 datasets. The best and second-best results are highlighted in **red** and **blue**, respectively. The *fps* values in the last column are generated on the same CPU using Pytorch as the backend. * indicates CPU-based *fps* value using ONNX-Runtime [18] (see Section 5.4.5).

the large-scale Youtube-BoundingBoxes dataset [95]. On this benchmark, the proposed *SMAT* outperforms all related trackers by at least 1.9% in *AUC*, 1.8% in P_{norm}, and 3% in *P*. No single tracker consistently exhibits the second-best performance.

- **LaSOT-test**: Results of the proposed *SMAT* on LaSOT-test show that our tracker has better long-term tracking performance than other lightweight trackers. The related E.T.Track employs a post-processing step to refine the predicted bounding boxes in order to reduce chances of target loss on long videos. Similarly, MixFormerV2, utilizes an online template update scheme to improve long-term tracking performance. Despite the absence of such bounding-box refinement step and template update scheme, our *SMAT* performs better than E.T.Track by 2.8% and MixFormerV2 by 0.7% in *AUC*.

- **NfS30:** This benchmark is created by downsampling the original NfS dataset (having 240 *fps* videos) to 30 *fps*, with simulated motion blur. The tracker results show that our *SMAT* is more resilient to motion blur than the related trackers.

- **UAV123**: With a better *AUC* by at least 0.9% than the related trackers, our *SMAT* demonstrates superior tracking on challenging videos under airborne scenarios.

- **AVisT:** The proposed tracker exhibits excellent results on recently published AVisT dataset, especially on the OP75 metric. With a higher *AUC* metric of 2.9% than the second-best HCAT, our *SMAT* showcases superior performance on videos with atmospheric adverse scenarios such as rain, fog, low-light, snow, and smoke.

- **TC128:** Our *SMAT* demonstrates a commendable performance on TC128 dataset, which pre-dominantly contains target objects of smaller size and low resolution, securing the second position with only a marginal difference of 0.2% from our *MVT* tracker (from Chapter 5).

- **OTB100:** This is one of the well-established tracking benchmarks, having 100 short-term videos. Our *SMAT* shows the best result across all lightweight trackers, while outperforming the second-best our *MVT* by a margin of 1.4%.

The consistent performance improvement across multiple datasets indicates the efficacy of employing CNNs and transformers for tracking. Notably, the concurrent deployment of these architectures in both the backbone and head modules contributes towards an accurate tracking model. This versatility enables effective tracking across diverse datasets, each presenting a unique set of challenges. The success of our approach reflects the benefits derived from combining the strengths of CNNs and transformers, showcasing the adaptability and performance of the proposed tracker across varied video scenarios.

The last column in Table 6.1 indicates the *fps* values of the proposed *SMAT* and related trackers, evaluated on a 12th Gen Intel(R) Core-i9 CPU. Leveraging the computational efficiency of separable self and mixed attention blocks in its model architecture, the proposed *SMAT* tracker achieves a real-time speed of 37 *fps*. Relative to standard transformer-based *MVT*, our *SMAT* is 25% faster as it uses transformers with separable attention in the backbone. While being faster than the fully transformer-based MixFormerV2-S by 19%, the proposed *SMAT* exhibits a lower *fps* value compared to related lightweight trackers with a two-stream pipeline [13, 121, 122, 127, 137]. This difference arises mainly due to the coupling of features in our tracker backbone, necessitating the computation of template and search region features at every frame during inference. However, by utilizing ONNX-Runtime [18] as the backend, we enhance the *fps* value of our tracker by 2× on CPU, achieving the best *fps* value compared to all related trackers. Additionally, we compute the *fps* values of our tracker on an Nvidia RTX 3090 GPU with PyTorch and TensorRT [134] as backends. Our tracker achieves nearly 1.5× speed-up in *fps* using TensorRT (239.7 *fps*) compared to PyTorch-based inference (158.1 *fps*).

Tracker	GOT10k-test		TrackingNet-test		#params ↓	*fps*	
	AO ↑	$SR_{0.50}$ ↑	*AUC* ↑	P_{norm} ↑	(in millions)	GPU ↑	CPU ↑
DiMP-50 [27]	0.611	0.717	0.740	0.801	26.1	61.5	15.0
Ocean [68]	0.615	0.735	0.693	0.794	44.3	127.8	10.1
TransT [28]	0.671	0.768	0.812	0.854	23.0	87.7	2.3
STARK-ST101 [13]	0.688	0.781	0.820	0.869	47.2	80.0	7.8
OSTrack-384 [14]	0.740	0.835	**0.839**	0.885	92.1	74.4	4.4
MixFormer-L [15]	**0.756**	**0.857**	**0.839**	**0.889**	183.9	45.2	< 5
MVT (ours)	0.627	0.723	0.756	0.816	5.5	**174.4**	29.5
SMAT (ours)	0.645	0.747	0.786	0.842	**3.8**	158.1	**36.9**

Table 6.2: Comparison of our *SMAT* and *MVT* with the state-of-the-art heavyweight trackers on server-based GOT10k and TrackingNet test datasets. Best and second-best results in accuracy and complexity (i.e., # of parameters and *fps*) are highlighted in red and blue.

6.4.3 Comparison to State-of-the-art Trackers

In Table 6.2, we present a comprehensive comparison between our proposed *SMAT* tracker, the state-of-the-art heavyweight trackers [13–15, 27, 28, 68], and our previously introduced *MVT* tracker from Chapter 5. While DiMP-50 [27] and Ocean [68] rely on purely CNN-based architectures, recent introduction of transformers for feature fusion [13,28] and backbone design [14,15], have assisted in achieving better performance for VOT. But, these improvements have increased the computational complexity (*e.g.*, higher number of model parameters) and lowered tracking speed. For the heavyweight trackers, we take the values of performance metrics from the respective papers; however, we compute their *fps* values on a GPU (i.e., Nvidia RTX 3090) and a CPU (i.e., 12th Gen Intel(R) Core-i9 processor), as shown in the last column of Table 6.2.

As the Table shows, proposed *SMAT* surpasses the popular trackers DiMP-50 and Ocean on GOT10k and TrackingNet datasets, demonstrating superior accuracy with fewer model parameters. It outperforms them in the *AUC* metric on average by 6.9% on the TrackingNet-test and 3.2% on the GOT10k-test, with a higher *fps* value on both GPU and CPU. Compared to our *MVT*, the proposed *SMAT* achieves a higher *AUC* scores on average across both datasets while exhibiting comparable computational complexity. Notably, both our proposed trackers run at significantly higher speeds on both CPU and GPU platforms compared to heavyweight OSTrack and MixFormer trackers, thereby providing a compelling trade-off between accuracy and computational efficiency for real-time tracking applications. For instance, although our *SMAT* has a lower *AUC* score than MixFormer-L by 8.2% across the two datasets, it is 48.4× more compact than MixFormer-L in terms of model parameters while being 7× faster on a CPU.

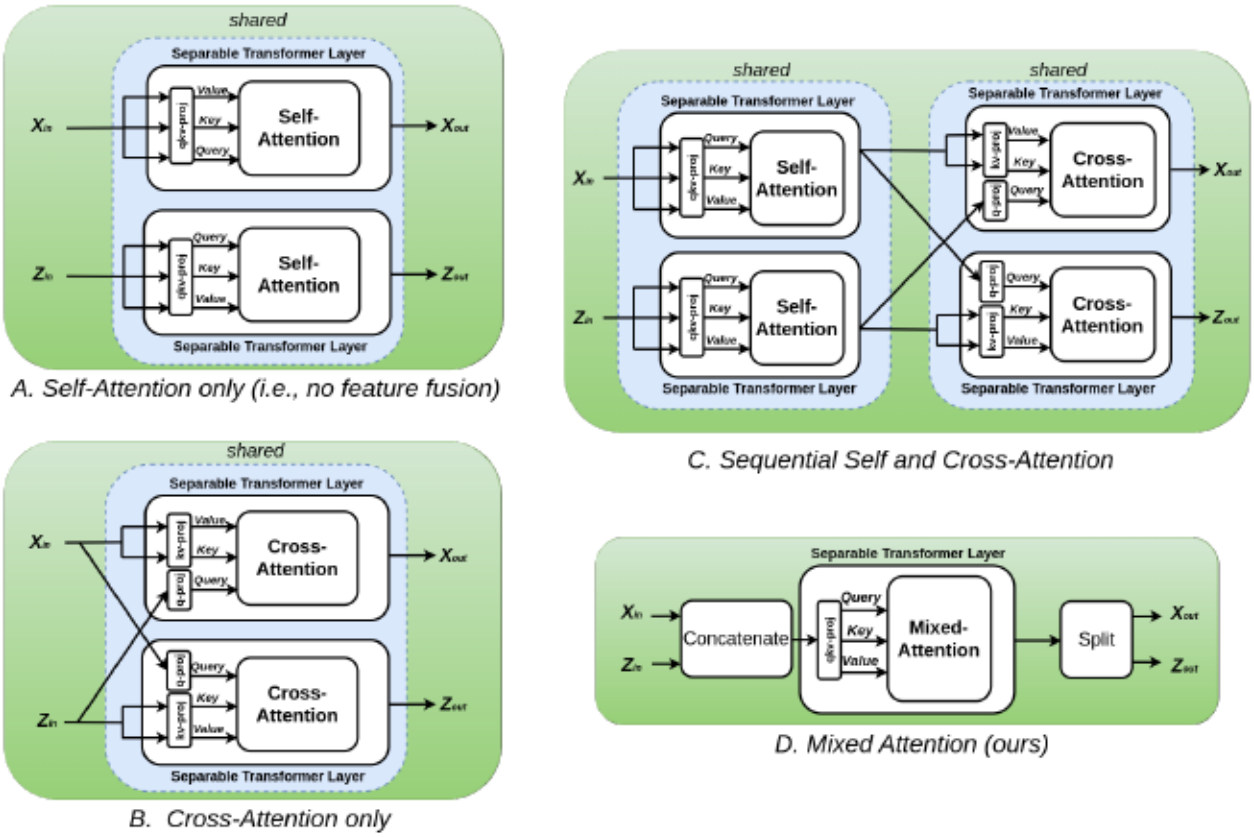

Figure 6.6: Comparing different feature fusion techniques (A, B and C) with the proposed mixed attention shown in D. Z_{in} and X_{in} denote the features corresponding to the target template and search regions, respectively.

6.4.4 Ablation Study

In this section, we present ablation study results quantifying the role of different components of our *SMAT* tracker. For this study, we use GOT10k-test [8] and LaSOT-test [9] due to their suitability towards gauging tracker generalization and long-term tracking performance, respectively.

Comparing Feature Fusion Techniques:

The proposed mixed attention efficiently approximates the explicit modeling of interaction *within* and *between* the target template and search regions. In this ablation study, we analyze the tradeoff between performance and latency of different attention mechanisms for feature fusion, depicted in Figure 6.6.

- **Variant-*A*** utilizes a shared separable self-attention transformer block for the two branches of the tracker backbone. This approach facilitates independent computation

Attention Mechanism	GOT10k-test [8]			LaSOT-test [9]			*fps*
	AO	$SR_{0.50}$	$SR_{0.75}$	*AUC*	P_{norm}	*P*	(CPU)
A	0.631	0.726	0.578	0.604	0.689	0.629	**39**
B	0.645	0.743	0.590	0.609	0.696	0.631	30
C	**0.654**	**0.761**	**0.598**	**0.621**	**0.717**	**0.656**	24
D (ours)	0.645	0.747	0.578	0.617	0.711	0.646	37

Table 6.3: Summarizing the feature fusion ablation study results for the proposed *SMAT* tracker. The best and second-best results are highlighted in **red** and blue, respectively.

of the template and search features for high tracking speed; however, it restricts the information exchange between the two regions (i.e., no feature fusion).

- **Variant-*B*** employs shared separable cross-attention transformers for inter-region feature fusion but excludes intra-region feature modeling. This approach is similar to the cross-feature blending for relation modeling by [28], which has shown excellent tracking results.

- **Variant-*C*** integrates cascaded self and cross-attention transformers, explicitly modeling both inter and intra-region feature fusion.

- **Variant-*D*** (Proposed) approximates explicit self and cross-attention computation by applying a separable self-attention on concatenated features.

We evaluate the performance of these variants on GOT10k and LaSOT test datasets summarized in Table 6.3. We can see that **Variant-*A*** has a 1.05× higher *fps* than the proposed method; however, the lack of information flow between the template and search region impacts its performance. Hence, compared to the proposed tracker, it has a lower *AUC* score by 1.4% and 1.3% on GOT10k and LaSOT, respectively. Performance of the pure cross-attention-based **Variant-*B*** is lower than our *SMAT* by 0.8% in *AUC* on the LaSOT dataset, while both approaches have comparable performance on the GOT10k dataset. However, **Variant-*B*** has a relatively lower *fps* by 18.9% than our *SMAT*, indicating that separate computation of cross-attention for the template and search regions is slower than our mixed-attention computation on concatenated features. **Variant-*C*** achieves the best results on both datasets; however, due to the explicit computation of self and cross-attention, it has the lowest *fps* value compared to the other variants and is relatively slower than our approach by 35% on a CPU. Comparatively, proposed **Variant-*D*** efficiently computes inter and intra-region feature fusion by using mixed-attention on the concatenated features and achieves a balanced trade-off in accuracy and speed. Our approach outperforms Variant-*A* on both datasets, while maintaining real-time tracking speed. It surpasses variant-*B* in

Predictor Head	GOT10k-test [8]			LaSOT-test [9]		
	AO	$SR_{0.50}$	$SR_{0.75}$	AUC	P_{norm}	P
Fully-Convolutional	0.610	0.709	0.540	0.594	0.682	0.612
Transformer-based (ours)	**0.645**	**0.747**	**0.578**	**0.617**	**0.711**	**0.646**

Table 6.4: Ablation study for the proposed predictor head. Best result is highlighted in **red**.

performance and speed. While variant-C achieves the best results, D provides a favorable compromise, offering the best overall trade-off between accuracy and speed.

Convolutional vs Transformer-based Head:

The ablation study quantifies the significance of the proposed separable self-attention transformer-based predictor head from Section 6.3.4. To assess its impact, we replace the proposed module with a fully-convolutional predictor head designed for classification and bounding-box regression. The results in Table 6.4 reveal that this substitution leads to a noticeable decrease in the performance of the proposed *SMAT* tracker. Specifically, there is a reduction of 3.5% in the AO metric for the GOT10k test dataset and 2.3% in AUC for the LaSOT test dataset. These findings highlight the significance of the proposed predictor head contributing to the overall tracking accuracy of the *SMAT* tracker.

Standard vs Separable Attention Mechanism:

To evaluate the efficiency of the separable attention mechanism integrated into our proposed *SMAT* tracker, we conducted an ablation study by comparing it against a variant employing the standard attention mechanism based on MobileViT [125]. The results in Table 6.5 demonstrate that proposed separable attention mechanism achieves competitive performance compared to the standard attention. Specifically, while the standard attention-based tracker exhibits a slight advantage in terms of Average Overlap (AO) on the GOT10k dataset, our separable attention-based approach achieves superior Area-Under-the Curve (AUC) on LaSOT dataset. Notably, despite the comparable performance, our proposed method with separable attention maintains a significant advantage in real-time speed, showcasing a 15.6% improvement over the standard attention-based tracker on CPU. This highlights the efficiency gains achieved by the separable attention mechanism in the proposed *SMAT* tracker.

Mixed Attention Mechanism	GOT10k-test [8]			LaSOT-test [9]			*fps* (CPU)
	AO	$SR_{0.50}$	$SR_{0.75}$	*AUC*	P_{norm}	*P*	
Standard	**0.659**	**0.760**	**0.605**	0.600	0.687	0.630	32
Separable (ours)	0.645	0.747	0.578	**0.617**	**0.711**	**0.646**	**37**

Table 6.5: Comparison of tracker performance using the standard *vs* separable mixed attention mechanism in the backbone. Best result is highlighted in **red**.

Tracker	*ARC*	*BC*	*CM*	*DEF*	*FM*	*FOC*	*IV*
LightTrack [121]	0.503	0.434	0.539	0.577	0.334	0.386	0.550
Stark-Lightning [13]	0.572	0.491	0.613	0.594	0.471	0.505	0.610
FEAR-XS [122]	0.488	0.437	0.528	0.505	0.389	0.403	0.506
HCAT [137]	0.587	0.524	0.639	0.619	0.460	0.507	0.606
E.T.Track [127]	0.573	0.526	0.590	0.619	0.404	0.480	0.612
MixFormerV2-S [138]	0.603	0.519	0.642	0.626	**0.507**	**0.539**	0.619
SMAT (ours)	**0.610**	**0.553**	**0.656**	**0.663**	0.466	0.517	**0.662**
Tracker	*LR*	*MB*	*OV*	*POC*	ROT	SV	*VC*
LightTrack [121]	0.407	0.457	0.441	0.497	0.519	0.523	0.502
Stark-Lightning [13]	0.516	0.568	0.557	0.554	0.577	0.582	0.581
FEAR-XS [122]	0.421	0.473	0.425	0.478	0.489	0.506	0.487
HCAT [137]	0.520	0.579	0.538	0.568	0.592	0.600	0.567
E.T.Track [127]	0.484	0.545	0.519	0.562	0.588	0.594	0.576
MixFormerV2-S [138]	**0.556**	**0.604**	**0.574**	0.586	0.603	0.617	**0.630**
SMAT (ours)	0.523	0.592	0.570	**0.600**	**0.621**	**0.624**	0.597

Table 6.6: Comparing the *AUC* values of the proposed *SMAT* with the related lightweight trackers for 14 attributes of the LaSOT dataset. The best and second-best results are highlighted in **red** and blue, respectively.

6.4.5 Attribute Analysis

The per-attribute analysis in Table 6.6 offers a detailed comparison of the proposed *SMAT* tracker with other lightweight trackers across various video challenges in the LaSOT dataset. Table 6.6 summarizes the tracker evaluation results across various attributes in the LaSOT dataset, including Aspect Ratio Change (*ARC*), Background Clutter (*BC*), Camera Motion (*CM*), Deformation (*DEF*), Fast Motion (*FM*), Full Occlusion (*FOC*), Illumination Variation (*IV*), Low Resolution (*LR*), Motion Blur (*MB*), Out-of-View (*OV*), Partial Occlusion (*POC*), Rotation (*ROT*), Scale Variation (*SV*), and Viewpoint Change (*VC*).

Examining the results in Table 6.6, it is evident that our tracker, *SMAT*, demonstrates superior performance on 8 out of 14 attributes, securing the top position. Additionally, in 5 attributes, our tracker achieves the second-best results. Notably, for the attributes

IV and *DEF*, *SMAT* outperforms the second-best tracker MixFormerV2-S [138] by a substantial margin, showcasing a higher *AUC* of 4.3% and 3.7%, respectively. When compared to related trackers that do not incorporate feature fusion in their backbone, such as [13,121,122,127,137], our tracker excels with a higher *AUC* of 4.4% for *DEF* and 2.3% for *ARC* compared to the second-best trackers, [127,137], and [137], respectively. These results highlight the efficacy of fusing features in our tracker backbone, enabling accurate bounding box predictions during significant target appearance variations. Furthermore, *SMAT* exhibits resilience to video challenges *POC* and *BC*, outperforming other trackers with a higher *AUC* of 3.2% and 2.7%, respectively. However, our tracker lags behind related trackers [138] and [13] in the *FM* attribute. Overall, the per-attribute analysis provides insights into the strengths and areas for improvement of the proposed *SMAT* tracker.

6.5 Summary

In this chapter, we proposed a novel separable self and mixed attention transformer-based architecture designed for lightweight tracking. The proposed backbone leverages a separable mixed attention transformer layer, facilitating the exchange of information between the target template and the search region. This results in superior feature encoding compared to traditional two-stream tracking pipelines. Furthermore, our separable self-attention transformer-based predictor head effectively captures long-range dependencies within the fused encoding, leading to superior target classification and accurate bounding-box predictions. An extensive ablation study was conducted to explore the accuracy-speed tradeoffs associated with various feature fusion methods. This study not only highlighted the effectiveness of our proposed head module in ensuring precise tracking but also quantified the efficiency of the separable mixed attention mechanism in terms of tracking speed. Our *SMAT* surpassed the performance of related lightweight trackers across eight challenging benchmarks. With a modest 3.8 million model parameters, our tracker achieved real-time speed on a CPU while reaching a speed of 158 *fps* on a GPU. Compared to state-of-the-art (heavyweight) trackers, our *SMAT* exhibits remarkable compromise between accuracy and efficiency in terms of model parameters and inference speed. For instance, although our *SMAT* has a lower *AUC* score than heavyweight MixFormer-L by 8.2% across GOT10k and TrackingNet datasets, it is 48.4 times more compact than MixFormer-L in terms of model parameters, while being 7× faster on a CPU.

Chapter 7

Conclusion and Future Work

7.1 Concluding Remark

This thesis adeptly addressed the challenge of enhancing the performance of visual object tracking (VOT) while effectively managing their computational complexity. The focus was primarily on two major paradigms in VOT algorithms: Discriminative Correlation Filter (DCF) and Siamese Network-based (SN) trackers.

For DCF tracking, the proposed online feature adaptation method effectively diminished the influence of non-discriminative features by transforming reliability scores derived from filter responses into feature weights. Recognizing the pivotal role of non-discriminative channels in target localization errors, we introduced a dynamic channel pruning method to mitigate their impact. These challenges are elegantly formulated as convex optimization problems, with prior knowledge of the strong correlation between successive frames encoded as the temporal regularizer. Our dual problem-based approach efficiently solved for adaptive weights to ensure high tracking speed, while providing a non-heuristic stopping criterion to guarantee convergence to the global minimum. To address the inherent limitation of rule-based reliability scores, we introduced an objective function that captures inter-channel relations, preventing false suppression of non-discriminative channels. Experimental verification of our approach by integrating them into popular DCF trackers showed significant enhancement in performance compared to the baseline and related DCF trackers.

Shifting the focus to SN trackers, we presented two algorithms: Mobile Vision Transformers (*MVT*) and Separable Self and Mixed Attention Transformer-based tracker (*SMAT*). These trackers addressed the necessity of modeling long-range feature dependencies within and between the target template and search regions for SN-based tracking, a facet notably absent in related CNN-based lightweight trackers. *MVT*, deploying Mobile Vision Transformers as its backbone with a hybrid arrangement of Convolutional Neural Network and

Transformer blocks, showcased superior feature encoding for target localization and outperforms lightweight counterparts. This hybrid design amalgamated the advantages of convolutions and transformers, ensuring fast tracking with fewer parameters compared to fully transformer-based backbone architectures. Further enhancing the efficiency of *MVT*, our *SMAT* replaced standard transformers with a separable attention block and introduced a CNN and transformer-based hybrid predictor head. Thorough evaluations on eight challenging benchmarks validated the effectiveness of the proposed SN tracking algorithms. *SMAT* emerged as a top-performer across all datasets, while *MVT* demonstrated competitive performance. Analyzing accuracy-speed tradeoffs against state-of-the-art (SOTA) trackers and conducting an insightful ablation study, we quantified the tradeoffs between different feature fusion methods and showcased the efficiency of separable mixed attention compared to standard attention-based tracking. Computational efficiency of the proposed architectures empowered our trackers to achieve real-time speed on a CPU and above 150 *fps* on a GPU. Additionally, we demonstrated that by leveraging frameworks such as ONNX-Runtime, it is possible to further increase the inference speed of our SN models on CPUs and GPUs, for example, from 36.9 to 74.1 *fps* on CPU.

Against the prevailing trend of increasing model parameters, we tackled the challenge of enhancing tracker performance with computationally efficient approaches. Our convex optimization-based solutions for DCF tracking demonstrated the potential for improving tracker performance without escalating computational complexity. By proposing a cascade of CNN and transformer blocks, we addressed the limitation of lacking feature fusion in the backbones of existing two-stream lightweight SN trackers, all while preserving high tracking speed. With the papers and codes accessible online, we hope that our contributions inspire future methods to prioritize both utility and accuracy in tracking algorithms.

7.2 Future Work

We categorize the possible extensions of our methods as follows.

- **Improving the tracker performance:** In Chapters 5 and 6, we have seen that the proposed combination of CNN and transformer modules has enabled our tracking methods to achieve superior results compared to related lightweight trackers. However, the qualitative results in Table 6.2 show that efficient VOT is a long way off matching the results of SOTA trackers. Therefore, we list a few directions of research that can potentially bridge the performance gap between SOTA and lightweight tracking methods.

– **Knowledge Distillation:** One potential avenue is leveraging Knowledge Distillation [139] to enhance the performance of lightweight trackers by harnessing the capabilities of a SOTA tracker. In this approach, the parameter-rich SOTA tracker, serves as a *teacher*, transferring its knowledge to a compact and lightweight *student* model. The aim is for the student model to mimic the teacher, capturing its informative features while remaining computationally efficient during tracking inference. Such a guided training, utilizing soft labels, has proven effective in creating compact models with improved generalization [74, 140].

– **Adaptive Local-Global Search Strategy:** Attribute analysis of the proposed *MVT* and *SMAT* trackers reveals challenges in videos categorized under *Fast Motion (FM)*, often involving small, fast-moving objects such as *volleyball* and *yo-yo*. Current SOTA trackers address these challenges by utilizing deep features and defining larger search areas to prevent target loss. However, our experimental results show that a simple increase in the search area heightens the sensitivity to distractor objects. We suspect that lack of discriminative features is the reason behind sucesptibility of lightweight tracking models to background clutter, thereby degrading the tracker performance. An intriguing direction for further exploration is adopting a local-global search strategy, which entails initiating a global search when tracking failures are detected [141–143]. Such an approach could enhance the robustness of lightweight trackers under challenges of tracking fast-moving objects, without the need for computationally expensive feature representations.

– **Integration of Natural Language Models:** The integration of natural language models into object tracking models has been a niche area of research [144–146]. However, with the rise of Large Language Models [147–150] and Foundation Models that integrate vision and language [151–154], this approach is gaining significant traction [155–161]. Leveraging these advancements towards creating effective ways of combining vision and language to improve VOT algorithms is an interesting direction to explore. Inspired by prompt-based interactions in large-language models such as ChatGPT, recent work has proposed a tracker model that receives text prompts to initialize the target object [162]. Eventhough [162] focuses on tracking pedestrians in videos, extending this idea to track generic objects and incorporating natural language prompts for correcting tracker drift enhances the utility of tracking algorithms, especially in video-editing applications. A further extension of these approaches towards speech-based input prompts could provide an optimal interactive interface between humans and

computers, ultimately enhancing the end-user experience.

- **Optimizing efficiency-related metrics:** While our approach has primarily focused on reducing the number of parameters to minimize model latency through fewer floating-point operations, it is also important to consider additional efficiency metrics, especially for practical deployment on embedded devices [163]. Energy efficiency becomes paramount in such scenarios, as excessive energy consumption can quickly drain the device's battery. Due to the challenges of measuring energy dissipation during CPU-based inference, our current work does not explicitly incorporate energy efficiency metrics. Beyond computational and energy efficiency, other factors contribute to model latency, including Memory Access Cost and Data I/O (i.e., reading and writing data from/to memory) [164]. Additionally, suboptimal implementations in the inference framework for certain model operations can further impact inference time. These bottlenecks are platform and framework-specific, making it challenging to design a one-size-fits-all architecture [165–167].

  Given the dependency of model latency on the deployment platform and the inference framework, Neural Architectural Search (NAS) [168] emerges as a promising avenue. NAS effectively considers diverse hardware and software constraints to design an optimal network [169–172]. It tailors the architecture to a specific device and framework, providing a dynamic solution that adapts to varied scenarios. NAS has demonstrated success in creating efficient modules for tracking purposes as well [121]. Therefore, incorporating NAS principles could further enhance the efficiency of tracking models, addressing specific constraints imposed by deployment platforms and frameworks.

- **Expanding the tracker functionality:** The existing VOT benchmarks predominantly define the task of target state estimation as predicting a rectangular bounding box encapsulating the target object. While widely accepted, an intriguing extension of this task involves predicting a precise segmentation mask as the tracker output [173]. Although more challenging, this extension holds significance, especially for applications related to video editing, such as video inpainting [174]. This approach offers a level of precision that can be valuable in reducing the manual intervention. While large-scale datasets for training such models are not readily available, recent attempts have utilized video object segmentation datasets [175] or created pseudo-labels to train segmentation-based tracking algorithms [176–178]. A notable challenge is the real-time execution of such segmentation-based tracking methods on a CPU, as existing solutions rely on GPUs for high-speed segmentation.

In the realm of tracking paradigms, much of the research is dedicated to either tracking single, generic objects or simultaneously tracking multiple objects belonging to specific object classes (e.g., humans) [20, 21]. Bridging these paradigms presents an exciting opportunity to develop a multi-object tracker capable of concurrently estimating trajectories for multiple generic objects. This task holds practical relevance for applications like video understanding and summarization, where simultaneous tracking of generic objects alleviates the computational burden associated with running separate single object tracking instances for each target object. With the emergence of recently published generic multi-object tracking datasets [179, 180], pursuing research in this direction opens new avenues for advancing the field.

Bibliography

[1] H. K. Galoogahi, T. Sim, and S. Lucey, "Multi-channel correlation filters," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[3] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022.

[4] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6638–6646, 2017.

[5] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, *et al.*, "The sixth Visual Object Tracking VOT2018 challenge results," in *European Conf. Computer Vision Workshops*, pp. 3–53, 2018.

[6] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, no. 9, pp. 1834–1848, 2015.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4510–4520, 2018.

[8] L. Huang, X. Zhao, and K. Huang, "GOT-10k: A large high-diversity benchmark for generic object tracking in the wild," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 43, no. 5, pp. 1562–1577, 2021.

[9] H. Fan, H. Bai, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, M. Huang, J. Liu, Y. Xu, *et al.*, "LaSOT: A high-quality large-scale single object tracking benchmark," *Int. J. Computer Vision*, vol. 129, pp. 439–461, 2021.

[10] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild," in *Proc. European Conf. Computer Vision*, pp. 300–317, 2018.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. European Conf. Computer Vision*, pp. 740–755, Springer, 2014.

[12] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1261–1274, 2016.

[13] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning spatio-temporal transformer for visual tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 10448–10457, 2021.

[14] B. Ye, H. Chang, B. Ma, S. Shan, and X. Chen, "Joint feature learning and relation modeling for tracking: A one-stream framework," in *Proc. European Conf. Computer Vision*, pp. 341–357, Springer, 2022.

[15] Y. Cui, C. Jiang, L. Wang, and G. Wu, "Mixformer: End-to-end tracking with iterative mixed attention," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 13608–13618, 2022.

[16] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning spatial-temporal regularized correlation filters for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4904–4913, 2018.

[17] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conf.*, BMVA Press, 2014.

[18] "ONNX Runtime: cross-platform, high performance ML inferencing and training accelerator." `https://github.com/microsoft/onnxruntime`. Accessed: 2023-10-15.

[19] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *Proc. European Conf. Computer Vision*, pp. 1–21, Springer, 2022.

[20] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 8844–8854, 2022.

[21] O. Cetintas, G. Brasó, and L. Leal-Taixé, "Unifying short and long-term tracking with graph hierarchies," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 22877–22887, 2023.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[23] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. European Conf. Computer Vision*, pp. 213–229, Springer, 2020.

[24] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. European Conf. Computer Vision*, pp. 850–865, Springer, 2016.

[25] S. Javed, M. Danelljan, F. S. Khan, M. H. Khan, M. Felsberg, and J. Matas, "Visual object tracking with discriminative filters and siamese networks: A survey and outlook," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 45, no. 5, pp. 6552–6574, 2023.

[26] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, no. 3, pp. 583–596, 2015.

[27] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 6182–6191, 2019.

[28] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 8126–8135, 2021.

[29] 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. 16743–16754, 2022.

[30] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 2544–2550, 2010.

[31] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 886–893, 2005.

[32] J. Van de Weijer, C. Schmid, J. Verbeek, and D. Larlus, "Learning color names for real-world applications," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1512–1523, 2009.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 770–778, 2016.

[36] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 3074–3082, 2015.

[37] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. European Conf. Computer Vision*, pp. 472–488, Springer, 2016.

[38] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 1401–1409, 2016.

[39] Y. Qi, S. Zhang, L. Qin, Q. Huang, H. Yao, J. Lim, and M.-H. Yang, "Hedging deep features for visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 41, no. 5, pp. 1116–1130, 2018.

[40] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, and H. Li, "Multi-cue correlation filters for robust visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4844–4853, 2018.

[41] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg, "Unveiling the power of deep tracking," in *Proc. European Conf. Computer Vision*, pp. 483–498, 2018.

[42] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 8, pp. 1561–1575, 2016.

[43] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conf. Computer Vision Workshops*, pp. 254–265, Springer, 2015.

[44] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. European Conf. Computer Vision*, pp. 702–715, Springer, 2012.

[45] S. Gladh, M. Danelljan, F. S. Khan, and M. Felsberg, "Deep motion features for visual tracking," in *Proc. IEEE Int. Conf. Pattern Recognition*, pp. 1243–1248, IEEE, 2016.

[46] M. Danelljan, G. Bhat, S. Gladh, F. S. Khan, and M. Felsberg, "Deep motion and appearance cues for visual tracking," *Elsevier Pattern Recognit. Lett.*, vol. 124, pp. 74–81, 2019.

[47] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 4310–4318, 2015.

[48] K. Dai, D. Wang, H. Lu, C. Sun, and J. Li, "Visual tracking via adaptive spatially-regularized correlation filters," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4670–4679, 2019.

[49] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler, "Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking," *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5596–5609, 2019.

[50] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler, "Joint group feature selection and discriminative filter learning for robust visual object tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 7950–7960, 2019.

[51] M. Mueller, N. Smith, and B. Ghanem, "Context-aware correlation filter tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 1396–1404, 2017.

[52] H. Kiani Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 1135–1143, 2017.

[53] Z. He, Y. Fan, J. Zhuang, Y. Dong, and H. Bai, "Correlation filters with weighted convolution responses," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, pp. 1992–2000, 2017.

[54] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6309–6318, 2017.

[55] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *Int. J. Computer Vision*, vol. 126, no. 7, pp. 671–688, 2018.

[56] G. Y. Gopal and M. A. Amer, "Reliable temporally consistent feature adaptation for visual object tracking," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pp. 2081–2085, IEEE, 2020.

[57] G. Y. Gopal and M. A. Amer, "Dynamic channel pruning for correlation filter based object tracking," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing (ICASSP)*, pp. 5700–5704, 2020.

[58] G. Y. Gopal and M. Amer, "Reliable interconnected channels for dynamic dcf based visual tracking," *Multimedia Tools and Applications*, vol. 83, no. 1, pp. 839–859, 2024.

[59] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2016.

[60] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of siamese visual tracking with very deep networks," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4282–4291, 2019.

[61] Z. Zhang and H. Peng, "Deeper and wider siamese networks for real-time visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4591–4600, 2019.

[62] S. Cheng, B. Zhong, G. Li, X. Liu, Z. Tang, X. Li, and J. Wang, "Learning to filter: Siamese relation network for robust tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4421–4431, 2021.

[63] B. Yan, X. Zhang, D. Wang, H. Lu, and X. Yang, "Alpha-refine: Boosting tracking performance by precise bounding box estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 5289–5298, 2021.

[64] Z. Zhang, Y. Liu, X. Wang, B. Li, and W. Hu, "Learn to match: Automatic matching network design for visual tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 13339–13348, October 2021.

[65] W. Han, X. Dong, F. S. Khan, L. Shao, and J. Shen, "Learning to fuse asymmetric feature maps in siamese trackers," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 16570–16580, 2021.

[66] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 8971–8980, 2018.

[67] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ATOM: Accurate tracking by overlap maximization," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4660–4669, 2019.

[68] Z. Zhang, H. Peng, J. Fu, B. Li, and W. Hu, "Ocean: Object-aware anchor-free tracking," in *Proc. European Conf. Computer Vision*, pp. 771–787, Springer, 2020.

[69] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese box adaptive network for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6668–6677, 2020.

[70] K. Yang, Z. He, W. Pei, Z. Zhou, X. Li, D. Yuan, and H. Zhang, "Siamcorners: Siamese corner networks for visual tracking," *IEEE Trans. Multimedia*, vol. 24, pp. 1956–1967, 2022.

[71] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proc. European Conf. Computer Vision*, pp. 101–117, 2018.

[72] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe, "Siam R-CNN: Visual tracking by re-detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2020.

[73] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic regression for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 7183–7192, 2020.

[74] J. Shen, Y. Liu, X. Dong, X. Lu, F. S. Khan, and S. Hoi, "Distilled siamese networks for visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 44, no. 12, pp. 8896–8909, 2021.

[75] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 2805–2813, 2017.

[76] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 1763–1771, 2017.

[77] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 4834–4843, 2018.

[78] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 1328–1338, 2019.

[79] J. Gao, T. Zhang, and C. Xu, "Graph convolutional tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2019.

[80] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, "SiamCAR: Siamese fully convolutional classification and regression for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6269–6277, 2020.

[81] Y. Yu, Y. Xiong, W. Huang, and M. R. Scott, "Deformable siamese attention networks for visual object tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6728–6737, 2020.

[82] Z. Zhou, W. Pei, X. Li, H. Wang, F. Zheng, and Z. He, "Saliency-associated object tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 9866–9875, 2021.

[83] D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, and C. Shen, "Graph attention tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 9543–9552, June 2021.

[84] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[85] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, and C. Schmid, "Vivit: A video vision transformer," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 6836–6846, 2021.

[86] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 16000–16009, 2022.

[87] 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. 16743–16754, 2022.

[88] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *Proc. European Conf. Computer Vision*, pp. 445–461, Springer, 2016.

[89] M. Noman, W. A. Ghallabi, *et al.*, "AVisT: A benchmark for visual object tracking in adverse visibility," in *British Machine Vision Conf.*, p. 817, BMVA Press, 2022.

[90] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 1125–1134, 2017.

[91] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, H. J. Chang, M. Danelljan, L. C. Zajc, A. Lukezic, *et al.*, "The tenth visual object tracking VOT2022 challenge results," in *Proc. European Conf. Computer Vision*, pp. 431–460, Springer, 2022.

[92] S. Gao, C. Zhou, and J. Zhang, "Generalized relation modeling for transformer tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 18686–18695, 2023.

[93] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: Algorithms and benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5630–5644, 2015.

[94] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2137–2155, Nov 2016.

[95] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 5296–5305, 2017.

[96] "TensorRT Open Source Software." `https://github.com/NVIDIA/TensorRT`. Accessed: 2023-10-15.

[97] G. Y. Gopal and M. A. Amer, "Mobile vision transformer-based visual object tracking," *British Machine Vision Conf.*, 2023.

[98] G. Y. Gopal and M. A. Amer, "Separable self and mixed attention transformers for efficient object tracking," in *IEEE Winter Conf. App. Computer Vision*, pp. 6708–6717, 2024.

[99] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090–1097, 2014.

[100] Z. Zhang, C. Wang, and X. Lu, "An adaptive feature channel weighting scheme for correlation tracking," in *International Conference on Artificial Neural Networks*, pp. 168–183, Springer, 2019.

[101] A. Bibi, M. Mueller, and B. Ghanem, "Target response adaptation for correlation filter tracking," in *Proc. European Conf. Computer Vision*, pp. 419–433, Springer, 2016.

[102] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming." accessed 20-October-2020.

[103] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[104] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[105] X. Lu, F. Tang, H. Huo, and T. Fang, "Learning channel-aware deep regression for object tracking," *Pattern Recognition Letters*, 2018.

[106] M. Che, R. Wang, Y. Lu, Y. Li, H. Zhi, and C. Xiong, "Channel pruning for visual tracking," in *European Conf. Computer Vision Workshops*, pp. 70–82, 2018.

[107] A. Lukezic, T. Vojir, L. ˇCehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *Int. J. Computer Vision*, vol. 126, no. 7, p. 671–688, 2018.

[108] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[109] M. Che, R. Wang, Y. Lu, Y. Li, H. Zhi, and C. Xiong, "Channel pruning for visual tracking," in *European Conf. Computer Vision Workshops*, pp. 70–82, 2018.

[110] C. Liu, P. Liu, W. Zhao, and X. Tang, "Visual tracking by structurally optimizing pre-trained CNN," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 30, no. 9, pp. 3153–3166, 2020.

[111] S. Ge, Z. Luo, C. Zhang, Y. Hua, and D. Tao, "Distilling channels for efficient deep tracking," *IEEE Trans. Image Process.*, vol. 29, pp. 2610–2621, 2020.

[112] X. An, Q. Liang, and N. Sun, "Multi-kernel support correlation filters with temporal filtering constraint for object tracking," *Multimedia Tools and Applications J.*, vol. 80, no. 9, pp. 14041–14073, 2021.

[113] X. Lu, C. Ma, B. Ni, and X. Yang, "Adaptive region proposal with channel regularization for robust object tracking," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 31, no. 4, pp. 1268–1282, 2021.

[114] T. Xu, Z. Feng, X.-J. Wu, and J. Kittler, "Adaptive channel selection for robust visual object tracking with discriminative correlation filters," *Int. J. Computer Vision*, pp. 1–17, 2021.

[115] M. Jain, A. Tyagi, A. V. Subramanyam, S. Denman, S. Sridharan, and C. Fookes, "Channel graph regularized correlation filters for visual object tracking," *IEEE Trans. Circuits Syst. Video Techn.*, pp. 1–1, 2021.

[116] F. Du, P. Liu, W. Zhao, and X. Tang, "Joint channel reliability and correlation filters learning for visual tracking," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 30, no. 6, pp. 1625–1638, 2020.

[117] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, p. 78–87, Oct. 2012.

[118] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[119] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng, "Efficient L_1 regularized logistic regression," in *Proc. 21st AAAI Conf. Artif. Intell.*, pp. 401–408, 2006.

[120] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, *et al.*, "The seventh visual object tracking VOT2019 challenge results," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, pp. 2206–2241, 2019.

[121] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, "Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 15180–15189, 2021.

[122] V. Borsuk, R. Vei, O. Kupyn, T. Martyniuk, I. Krashenyi, and J. Matas, "FEAR: Fast, efficient, accurate and robust visual tracker," in *Proc. European Conf. Computer Vision*, pp. 644–663, Springer, 2022.

[123] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 3101–3109, IEEE, 2015.

[124] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing convolutions to vision transformers," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 22–31, 2021.

[125] S. Mehta and M. Rastegari, "MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer," in *International Conference on Learning Representations*, 2022.

[126] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "DetNAS: Backbone search for object detection," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[127] P. Blatter, M. Kanakis, M. Danelljan, and L. Van Gool, "Efficient visual tracking with exemplar transformers," in *IEEE Winter Conf. App. Computer Vision*, pp. 1571–1581, 2023.

[128] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style convnets great again," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 13733–13742, 2021.

[129] Z. Cao, C. Fu, J. Ye, B. Li, and Y. Li, "HiFT: Hierarchical feature transformer for aerial tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 15457–15466, 2021.

[130] J. Dai, Y. Fu, S. Wang, and Y. Chang, "Siamese hierarchical feature fusion transformer for efficient tracking," *Frontiers in Neurorobotics*, 2022.

[131] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

[132] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[133] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[134] "NVIDIA TensorRT." `https://developer.nvidia.com/tensorrt`. Accessed: 2023-10-15.

[135] S. Mehta and M. Rastegari, "Separable self-attention for mobile vision transformers," *Transactions on Machine Learning Research*, 2023.

[136] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *CoRR*, vol. abs/2006.04768, 2020.

[137] X. Chen, B. Kang, D. Wang, D. Li, and H. Lu, "Efficient visual tracking via hierarchical cross-attention transformer," in *Proc. European Conf. Computer Vision*, pp. 461–477, Springer, 2022.

[138] Y. Cui, T. Song, G. Wu, and L. Wang, "Mixformerv2: Efficient fully transformer tracking," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[139] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[140] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Computer Vision*, vol. 129, pp. 1789–1819, 2021.

[141] B. Yan, H. Zhao, D. Wang, H. Lu, and X. Yang, "Skimming-perusal tracking: A framework for real-time and robust long-term tracking," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 2385–2393, 2019.

[142] X. Wang, J. Tang, B. Luo, Y. Wang, Y. Tian, and F. Wu, "Tracking by joint local and global search: A target-aware attention-based approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6931–6945, 2021.

[143] H. Zhao, B. Yan, D. Wang, X. Qian, X. Yang, and H. Lu, "Effective local and global search for fast long-term tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 45, no. 1, pp. 460–474, 2022.

[144] Q. Feng, V. Ablavsky, Q. Bai, G. Li, and S. Sclaroff, "Real-time visual object tracking with natural language description," in *IEEE Winter Conf. App. Computer Vision*, pp. 700–709, 2020.

[145] X. Wang, X. Shu, Z. Zhang, B. Jiang, Y. Wang, Y. Tian, and F. Wu, "Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 13763–13773, 2021.

[146] Q. Feng, V. Ablavsky, Q. Bai, and S. Sclaroff, "Siamese natural language tracker: Tracking by natural language descriptions with siamese trackers," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 5851–5860, 2021.

[147] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1*, pp. 4171–4186, Association for Computational Linguistics, 2019.

[148] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[149] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[150] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[151] X. Zou, Z.-Y. Dou, J. Yang, Z. Gan, L. Li, C. Li, X. Dai, H. Behl, J. Wang, L. Yuan, *et al.*, "Generalized decoding for pixel, image, and language," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 15116–15127, 2023.

[152] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, pp. 8748–8763, PMLR, 2021.

[153] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23716–23736, 2022.

[154] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International Conference on Machine Learning*, pp. 12888–12900, PMLR, 2022.

[155] Y. Zheng, B. Zhong, Q. Liang, G. Li, R. Ji, and X. Li, "Towards unified token learning for vision-language tracking," *IEEE Trans. Circuits Syst. Video Techn.*, 2023.

[156] M. Guo, Z. Zhang, H. Fan, and L. Jing, "Divert more attention to vision-language tracking," in *Advances in Neural Information Processing Systems* (A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds.), 2022.

[157] C. Zhang, X. Sun, Y. Yang, L. Liu, Q. Liu, X. Zhou, and Y. Wang, "All in one: Exploring unified vision-language tracking with multi-modal alignment," in *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 5552–5561, 2023.

[158] X. Li, Y. Huang, Z. He, Y. Wang, H. Lu, and M.-H. Yang, "Citetracker: Correlating image and text for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 9974–9983, 2023.

[159] H. Zhang, J. Wang, J. Zhang, T. Zhang, and B. Zhong, "One-stream vision-language memory network for object tracking," *IEEE Transactions on Multimedia*, 2023.

[160] D. Ma and X. Wu, "Tracking by natural language specification with long short-term context decoupling," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 14012–14021, 2023.

[161] R. Wang, Z. Tang, Q. Zhou, X. Liu, T. Hui, Q. Tan, and S. Liu, "Unified transformer with isomorphic branches for natural language tracking," *IEEE Trans. Circuits Syst. Video Techn.*, 2023.

[162] P. Nguyen, K. G. Quach, K. Kitani, and K. Luu, "Type-to-track: Retrieve any object via prompt-based tracking," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[163] H. Tabani, A. Balasubramaniam, E. Arani, and B. Zonooz, "Challenges and obstacles towards deploying deep learning models on mobile devices," *arXiv preprint arXiv:2105.02613*, 2021.

[164] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proc. European Conf. Computer Vision*, pp. 116–131, 2018.

[165] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 2820–2828, 2019.

[166] L. L. Zhang, Y. Yang, Y. Jiang, W. Zhu, and Y. Liu, "Fast hardware-aware neural architecture search," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 692–693, 2020.

[167] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "Hardware-aware neural architecture search: Survey and taxonomy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (Z.-H. Zhou, ed.), pp. 4322–4329, International Joint Conferences on Artificial Intelligence Organization, 8 2021. Survey Track.

[168] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 8697–8710, 2018.

[169] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International Conference on Machine Learning*, pp. 10096–10106, PMLR, 2021.

[170] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 10734–10742, 2019.

[171] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, *et al.*, "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 12965–12974, 2020.

[172] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia, *et al.*, "Chamnet: Towards efficient network design through platform-aware model adaptation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 11398–11407, 2019.

[173] M. Kristan, J. Matas, M. Danelljan, M. Felsberg, H. J. Chang, L. Č. Zajc, A. Lukežič, O. Drbohlav, Z. Zhang, K.-T. Tran, *et al.*, "The first visual object tracking segmentation VOTS2023 challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1796–1818, 2023.

[174] Z. Li, C.-Z. Lu, J. Qin, C.-L. Guo, and M.-M. Cheng, "Towards an end-to-end framework for flow-guided video inpainting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17562–17571, 2022.

[175] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 DAVIS challenge on video object segmentation," *arXiv preprint arXiv:1704.00675*, 2017.

[176] A. Lukezic, J. Matas, and M. Kristan, "D3S-a discriminative single shot segmentation tracker," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 7133–7142, 2020.

[177] M. Paul, M. Danelljan, C. Mayer, and L. Van Gool, "Robust visual tracking by segmentation," in *Proc. European Conf. Computer Vision*, pp. 571–588, Springer, 2022.

[178] W. Hu, Q. Wang, L. Zhang, L. Bertinetto, and P. H. Torr, "Siammask: A framework for fast online object tracking and segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 45, no. 3, pp. 3072–3089, 2023.

[179] H. Bai, W. Cheng, P. Chu, J. Liu, K. Zhang, and H. Ling, "GMOT-40: A benchmark for generic multiple object tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 6719–6728, 2021.

[180] C. Mayer, M. Danelljan, M.-H. Yang, V. Ferrari, L. Van Gool, and A. Kuznetsova, "Beyond SOT: Tracking multiple generic objects at once," in *IEEE Winter Conf. App. Computer Vision*, pp. 6826–6836, 2024.

Appendix A

Derivation of the dual problem for feature adaptation

This supplementary material derives the dual for the minimization problem (or the primal problem) defined in Eq. 11a from Chapter 2. We begin by defining the Lagrangian of primal problem as

$$L(\boldsymbol{\alpha}, \mathbf{a}, b) = -\sum_{i=1}^{M} \alpha_i r_i + \frac{\gamma}{2} \sum_{i=1}^{M} (\alpha_i - \beta_i)^2 - \sum_{i=1}^{M} a_i \alpha_i + b \left(\sum_{i=1}^{M} \alpha_i - 1 \right) \tag{35}$$

where $a_i \in R_{\geq 0}$ for $i = \{1, \ldots, M\}$ and $b \in R$ are the dual variables corresponding to the constraints in Eq. 11b and Eq. 11c, respectively. Eq. 35 can be re-written in the vectorized form as

$$L(\boldsymbol{\alpha}, \mathbf{a}, b) = -\langle \boldsymbol{\alpha}, \mathbf{r} \rangle + \frac{\gamma}{2} \|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2^2 - \langle \boldsymbol{\alpha}, \mathbf{a} \rangle + b \left(\langle \boldsymbol{\alpha}, \mathbb{1} \rangle - 1 \right) \tag{36}$$

$$= \frac{\gamma}{2} \|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2^2 + \langle \boldsymbol{\alpha}, -\mathbf{r} - \mathbf{a} + b \cdot \mathbb{1} \rangle - b \tag{37}$$

From Eq. 36, the dual Lagrangian is defined as,

$$\mathrm{q}(\mathbf{a}, b) = \inf_{\boldsymbol{\alpha} \in R^M} L(\boldsymbol{\alpha}, \mathbf{a}, b). \tag{38}$$

To derive $q(\mathbf{a}, b)$, we compute $\nabla_{\boldsymbol{\alpha}} L$ as

$$\nabla L_{\boldsymbol{\alpha}} = \gamma \left(\boldsymbol{\alpha} - \boldsymbol{\beta} \right) + \left(-\mathbf{r} - \mathbf{a} + b \cdot \mathbb{1} \right). \tag{39}$$

By setting $\nabla_{\boldsymbol{\alpha}} L = 0$, we obtain an expression for the primal variable $\boldsymbol{\alpha}$ as

$$\boldsymbol{\alpha} = \boldsymbol{\beta} - \frac{1}{\gamma}\left(-\mathbf{r} - \mathbf{a} + b \cdot \mathbb{1}\right). \tag{40}$$

We eliminate the primal variable $\boldsymbol{\alpha}$ from Eq. 36 by substituting $\boldsymbol{\alpha}$ with the expression in Eq. 40. Finally, we obtain the dual problem

$$\underset{\mathbf{a}, b}{\text{maximize}} \quad -\frac{1}{2\gamma}\left\|-\mathbf{r} - \mathbf{a} + b \cdot \mathbb{1}\right\|_2^2 + \langle \boldsymbol{\beta}, -\mathbf{r} - \mathbf{a} + b \cdot \mathbb{1} \rangle - b \tag{41a}$$

$$\text{subject to} \quad \mathbf{a} \geq 0. \tag{41b}$$

The constraint in Eq. 41b exists since the dual variable $\mathbf{a}$ is associated with the inequality constraints defined in Eq. 11b.

We can see that the dual objective is concave in dual variables $\mathbf{a}$ and b, whereas the constraints are decoupled unlike the primal problem in Eq. 11a. We use cyclic coordinate ascent to solve for dual variables $\mathbf{a}$ and b, as described in Section 2.3.4. We use the condition of strong duality to derive the stopping criterion, described in Algorithm 1.

Appendix B

Derivation of the dual problem for dynamic channel pruning

This supplementary material derives the dual for the optimization problem defined in Eq. 17 from Chapter 3. We form the Lagrangian of primal problem in Eq. 17 as,

$$L(\boldsymbol{\alpha}, \mathbf{a}, b) = -\frac{1}{N}\langle \boldsymbol{\alpha}, \mathbf{r} \rangle + \frac{\gamma_1}{2} \left\| \boldsymbol{\alpha} - \boldsymbol{\beta} \right\|_2^2 + \frac{\gamma_2}{N}\langle \mathbb{1}, \boldsymbol{\rho} \rangle - \langle \mathbf{a}, \boldsymbol{\alpha} \rangle + \langle b, \boldsymbol{\alpha} - \mathbb{1} \rangle + \langle \mathbf{c}_1, -\boldsymbol{\alpha} - \boldsymbol{\rho} \rangle + \langle \mathbf{c}_2, \boldsymbol{\alpha} - \boldsymbol{\rho} \rangle + d \cdot (p \cdot N - \langle \mathbb{1}, \boldsymbol{\alpha} \rangle) \quad (42)$$

where $\{\mathbf{a}, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2\} \in R^N_{\geq 0}$ and $d \in R_{\geq 0}$ are the dual variables. Non-negativity constraint exist for the dual variables since all of them are associated with inequality constraints. The above Lagrangian can be re-written as

$$L(\boldsymbol{\alpha}, \mathbf{a}, b) = \frac{\gamma_1}{2} \left\| \boldsymbol{\alpha} - \boldsymbol{\beta} \right\|_2^2 + \left\langle \boldsymbol{\alpha}, \left(-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} - \mathbf{c}_1 + \mathbf{c}_2 - d \cdot \mathbb{1} \right) \right\rangle + \left\langle \boldsymbol{\rho}, \left(\frac{\gamma_2}{N} \cdot \mathbb{1} - \mathbf{c}_1 - \mathbf{c}_2 \right) \right\rangle - \langle \mathbf{b}, \mathbb{1} \rangle + d \cdot p \cdot N. \quad (43)$$

From Eq. 43, we define the dual Lagrangian as,

$$\mathrm{q}(\mathbf{a}, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2, d) = \inf_{\{\boldsymbol{\alpha}, \boldsymbol{\rho}\} \in R^N} L(\boldsymbol{\alpha}, \boldsymbol{\rho}, \mathbf{a}, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2, d). \quad (44)$$

To derive $\mathrm{q}(\mathbf{a}, \mathbf{b}, \mathbf{c}_1, \mathbf{c}_2, d)$, we compute $\nabla_{\boldsymbol{\alpha}} L$ and $\nabla_{\boldsymbol{\rho}} L$ as,

$$\nabla L_{\boldsymbol{\alpha}} = \gamma \left(\boldsymbol{\alpha} - \boldsymbol{\beta} \right) + \left(-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} - \mathbf{c}_1 + \mathbf{c}_2 - d \cdot \mathbb{1} \right), \quad (45)$$

$$\nabla L_{\boldsymbol{\rho}} = \left(\frac{\gamma_2}{N} \cdot \mathbb{1} - \mathbf{c}_1 - \mathbf{c}_2\right). \tag{46}$$

Next, we set $\nabla L_{\boldsymbol{\alpha}}$ and $\nabla L_{\boldsymbol{\rho}}$ to zero to obtain the following conditions,

$$\boldsymbol{\alpha} = \boldsymbol{\beta} - \frac{1}{\gamma_1}\left(-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} - \mathbf{c}_1 + \mathbf{c}_2 - d \cdot \mathbb{1}\right), \tag{47}$$

$$\frac{\gamma_2}{N} \cdot \mathbb{1} - \mathbf{c}_1 - \mathbf{c}_2 = 0. \tag{48}$$

We substitute $\boldsymbol{\alpha}$ from Eq. 47 in the Lagrangian from Eq. 43 to obtain the dual problem,

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{b},\mathbf{c}_1,\mathbf{c}_2,d}{\text{maximize}} \quad & -\frac{1}{2\gamma_1}\left\|-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} - \mathbf{c}_1 + \mathbf{c}_2 - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right\|_2^2 - \langle \mathbb{1}, b\rangle + d \cdot p \cdot N \\
& + \left\langle \boldsymbol{\beta}, -\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} - \mathbf{c}_1 + \mathbf{c}_2 - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right\rangle \qquad (49) \\
\text{subject to} \quad & 0 \leq \mathbf{a}, \\
& 0 \leq \mathbf{b}, \\
& 0 \leq \mathbf{c}_1, \\
& 0 \leq \mathbf{c}_2, \\
& 0 \leq d, \\
& \frac{\gamma_2}{N} \cdot \mathbb{1} - \mathbf{c}_1 - \mathbf{c}_2 = 0.
\end{aligned}$$

We eliminate the variable $\mathbf{c}_1$ from the dual problem by substituting $\mathbf{c}_1 = \frac{\gamma_2}{N} \cdot \mathbb{1} - \mathbf{c}_2$ in Eq. 49 to obtain an equivalent maximization problem

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{b},\mathbf{c}_2,d}{\text{maximize}} \quad & -\frac{1}{2\gamma_1}\left\|-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} + 2\mathbf{c}_2 - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right\|_2^2 \\
& + \left\langle \boldsymbol{\beta}, -\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} + 2\mathbf{c}_2 - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right\rangle - \langle \mathbb{1}, b\rangle + d \cdot p \cdot N \qquad (50) \\
\text{subject to} \quad & 0 \leq \mathbf{a}, \\
& 0 \leq \mathbf{b}, \\
& 0 \leq \mathbf{c}_2 \leq \frac{\gamma_2}{N}\mathbb{1}, \\
& 0 \leq d.
\end{aligned}$$

Constraint $0 \leq \mathbf{c}_2 \leq \frac{\gamma_2}{N}\mathbb{1}$ guarantees that we do not violate the non-negativity constraint associated with the eliminated variable $\mathbf{c}_1$. Also, we use the variable $\mathbf{c}$ instead of $\mathbf{c}_2$ in Eq. 18 for mathematical convenience.

We can see that the dual objective in Eq. 50 is concave in dual variables $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}_2$, and d. To solve for the dual variables, we use cyclic block coordinate ascent as described in Section 3.3.2. Finally, we compute the channel weights using the dual variables as

$$\boldsymbol{\alpha} = \boldsymbol{\beta} - \frac{1}{\gamma_1}\left(-\frac{1}{N}\mathbf{r} - \mathbf{a} + \mathbf{b} + 2\mathbf{c}_2 - \left(\frac{\gamma_2}{N} + d\right)\mathbb{1}\right). \tag{51}$$