IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.DOI

Siamese Visual Object Tracking: A Survey

MILAN ONDRAŠOVIČ[1], PETER TARÁBEK[1]
[1]Department of Mathematical Methods and Operations Research, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovakia
Corresponding author: Milan Ondrašovič (e-mail: milan.ondrasovic@fri.uniza.sk).

This work was supported by the research grant VEGA 1/0689/19.

ABSTRACT Object tracking belongs to active research areas in computer vision. We are interested in matching-based trackers exploiting deep machine learning known as Siamese trackers. Their powerful capabilities stem from similarity learning. This tracking paradigm is promising due to its inherent balance between performance and efficiency, so trackers of this type are suitable for real-time generic object tracking. There is an upsurge in research interest in Siamese trackers and the lack of available specialized surveys in this category. In this survey, we aim to identify and elaborate on the most significant challenges the Siamese trackers face. Our goal is to answer what design decisions the authors made and what problems they attempted to solve in the first place. We thus perform an in-depth analysis of the core principles on which Siamese trackers operate with a discussion of incentives behind them. Besides, we provide an up-to-date qualitative and quantitative comparison of the prominent Siamese trackers on established benchmarks. Among other things, we discuss current trends in developing Siamese trackers. Our survey could help absorb the details about the underlying principles of Siamese trackers and the challenges they face.

INDEX TERMS visual object tracking, deep learning, Siamese neural networks, similarity learning, fully convolutional networks

I. INTRODUCTION

OBJECT tracking is among very active research areas in the field of computer vision [1]. Generally speaking, object tracking is a fundamental task where some degree of reasoning about a specific set of objects is required to establish an object correspondence between frames [2]. In this work, we consider the task of visual object tracking (VOT) in a video. When evaluating the performance on this task, the object of interest is identified solely using an axis-aligned bounding box (BBOX) in the first frame. The tracking algorithm should then preserve the assigned object's identity in future frames.

Despite the plethora of literature [3], [4], [5], the task remains a challenge due to changes in object appearance caused by scale and lighting variation, deformation, rotation, occlusion, and background clutter [6]. The potential for real-world application is vast, ranging from video surveillance [7] and traffic analysis [8], through human-computer interaction [9] to robotics [10] and even video compression standards [11].

During the process of VOT, a major challenge is to properly discriminate the target object from the background as well as from the other objects. Because of this, we can separate the existing appearance-based trackers into discriminative and generative. Discriminative approaches treat VOT as a binary classification problem. The aim is to separate the foreground from the background. In generative models, candidates are searched to minimize reconstruction errors [5], [12]. In the past, these models relied on low-level, handcrafted features. The primary limitations of those approaches were incapability to capture semantic features and not being robust to considerable appearance variations [13]. However, deep machine learning is an excellent tool for powerful feature extraction [14]. Features learned by convolutional neural networks (CNNs) carry rich semantic information [15]. Models based on deep learning are strong at distinguishing objects of different categories with good generalization capabilities [13]. With this in mind, the key is to find features that simultaneously allow differentiating between an object and a background and allow handling changes of the tracked object, even when not known a priori [16]. In this work, we deal with trackers that exploit the generative approach.

Currently, popular visual tracking methods revolve around Siamese neural networks (section III). The Siamese-based networks are considered the most promising architectures

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

based on their balance between performance and efficiency [17]. With this in mind, these architectures are the primary focus of this survey.

Even though the Siamese models were first utilized for signature verification [18], they can also be exploited for VOT by formulating it as a target matching problem. Simply put, a Siamese tracker accepts two input patches, an exemplar image (what to look for), and a search image (where to look). It is a Y-shaped network that joins two branches to produce a single output [1], [19]. The tracker attempts to localize the exemplar (target template) provided at the beginning in the search regions contained in future frames. The goal, therefore, is to learn a general similarity mapping between the exemplar and the search region [20] (section III-A).

Many of the recent Siamese trackers are also fully convolutional [12]. Among other things, this allows them to handle variable dimensions of input images. We emphasize this property because we mention it many times as it makes the development of architectures with fewer hyper-parameters and allows end-to-end training [21] (section III-B). Modern approaches for object detection [21] and segmentation [22] go in the fully convolutional direction, too. Among other things, it is well-known within the tracking community that the fully convolutional trackers can effectively capture translation, which is the main variation in the video [23].

Since the field of VOT is enormous, we had to narrow our focus down to the specialized type of trackers. A narrow scope was crucial since we wanted to target individual traits of specific frameworks. We do not deeply elaborate on solutions that utilize information other than visual. Moreover, we only address generic single object tracking. The literature on deep learning-based VOT has been growing steadily since 2015, so it is hard to compose a broad survey about such a vast body of publications. In this work, we strive for deep analysis of a given tracking paradigm rather than to provide a comprehensive, broad discussion covering the vast population of approaches to object tracking. For this purpose, other works complement our contribution (section II).

Siamese-based tracking is relatively new, and despite numerous advantages, it still has limitations. The main purpose of this survey is to convey the main properties of Siamese trackers. Besides, it also provides a discussion about object tracking in general. We aim at reflecting on ways by which researchers have attempted to reinforce the pros and suppress the cons (section IV). Therefore, our **contribution** is:

- This work provides a specialized discussion about inherent traits regarding Siamese trackers. We perform an in-depth analysis of the core principles of Siamese tracking.
- We dissect the current issues with the Siamese architectures and analyze the incentives behind the building blocks the authors employed to address them.
- We offer an up-to-date quantitative and qualitative comparison of the surveyed trackers on the established benchmarks.

This work provides a description of Siamese frameworks to help spot their common strengths and weaknesses to address them appropriately. Therefore, it may be helpful when utilizing existing or designing new Siamese trackers.

The rest of the paper is organized as follows. The upcoming section II summarizes similar works related to surveying object tracking, especially Siamese-based trackers. Section III describes the problem of VOT using similarity learning. Additionally, we describe the fundamental components of Siamese architecture. The subsequent section IV aims at the current prominent traits and limitations of Siamese tracking and summarizes how current works have attempted to resolve them. In section V, we discuss experimental results of the surveyed trackers on standard benchmarks together with existing trends in their use. In the last section VI, we sum up our findings and highlight potentially important directions in future research.

II. RELATED WORK

Yilmaz et al. [4] in 2006 created one of the deepest and very comprehensive surveys on object tracking. This work is pertinent to our survey mainly due to the discussion regarding object representation. They noted the importance of object detection, which inherently faces similar problems as tracking itself. Other considered topics were motion prediction and foreground segmentation, both of which are sometimes exploited in tracking.

The most recent comprehensive survey is from 2021 by Marvasti-Zadeh et al. [17]. This work broadly covers existing approaches to object tracking in general. To the best of our knowledge, this paper covers the greatest number of deep learning-based trackers developed since 2013. Their contributions were establishing a taxonomy of trackers, highlighting the current issues and proposed solutions, comparing available datasets by various properties, and finally, extensive experimental evaluations of trackers on numerous benchmark datasets. They also discerned the chosen trackers according to their core principles and architectural components. Among other things, their work also includes Siamese trackers. This survey is very extensive and a helpful guide to equip the reader with current trends in VOT to understand the broad picture. However, we do differ in our approaches. In our case, we focus specifically on Siamese-based tracking, so we provide more detailed descriptions of a specific subset of trackers. Our goal was to go in-depth rather than breadth. We strove to dissect the essential components of Siamese trackers along with the problems that the designers were trying to solve.

As we will point out later, Siamese trackers are better at discerning foreground from the background than between objects of similar appearance. A survey from [24] discusses these possibilities. Furthermore, they also introduced similarity measures. This aspect is necessary for similarity learning, an important building block of Siamese networks. Another comprehensive survey on Siamese tracking was done by Pflugfelder [1]. His paper covers general challenges of VOT,

tracker design, and representative Siamese architectures. He also described details of Siamese trackers such as loss functions, architectural components, design choices, and training data. A relevant contribution of this work is the development of a Lisp-like functional formalism to express Siamese architectures for a general comparison of their structure. One of his conclusions was that the evaluation methodology of trackers back then had several flaws. Reproduction of results and their comparison was difficult to achieve accurately. We identify this survey as complementary to ours. However, our perspective is different. We put general problems with Siamese trackers at the core and then discuss how architectures address them. Other surveys, including [1], provide an overview of "how" the trackers operate with respective building blocks. Conversely, we elaborate on "what" obstacles the trackers faced and what architectural decisions the authors made to tackle them. Additionally, we also cover new advancements since the year 2017.

Pflugfelder [1] also remarked that datasets are necessary for successful VOT. The work of [25] contains a survey of datasets for visual tracking. Another survey of datasets is [26], where the authors also discuss challenging features of established benchmark datasets. A recent, broad dissection of object tracking datasets is from Marvasti-Zadeh et al. [17]. They thoroughly compared VOT benchmark datasets on their fundamentals characteristics, such as the number of frames, videos, and classes; various sequence attributes; object classes; and overlaps with other datasets.

A survey by [27] provides a general description of visual tracking together with its challenges. Apart from this, it also provides a list of visual feature descriptors followed by a discussion of trackers using deep learning. They partially cover Siamese trackers, too. Nevertheless, they aimed to propose a survey on the usability of tracking frameworks in mobile robots. Since visual features and appearance models are relevant to well-performing trackers, Li et. al [5] did an extensive survey on appearance models in VOT. This work focuses on traditional computer vision approaches not based on machine learning. They discuss topics such as feature extraction, optical flow, Gaussian mixture model (GMM), to mention a few. We do consider their elaboration important to our research. Various Siamese trackers utilize some of those methods (e.g., optical flow in [28], GMM in [29]), apart from deep learning. One of the leading causes of difficulties for VOT is a variation of object appearance [6]. In this regard, there is a survey on adaptive visual representations in tracking [30].

Siamese trackers inherently exploit the capabilities of deep machine learning. A short survey concerning deep learning-based trackers is [31]. A recent, relevant survey paper from [32] covers online learning methods for visual tracking. They also cover general challenges of VOT, CNN-based trackers, multiple perspectives on the classification of trackers, evaluation metrics, and a broad review of Siamese trackers. This survey is significant for the general understanding of deep learning-based object tracking. Fiaz et al. [33] compiled a thorough analysis of hand-crafted as well as deep learning-based trackers, including the Siamese ones. They experimentally evaluated the robustness of different trackers. Our work also briefly mentions a taxonomy of tracking algorithms. In the case of Siamese, inspired by [34], they established three categories: early, intermediate, and late merge concerning how they process input and the extracted features.

III. SIAMESE TRACKING

Deep Siamese neural networks are non-linear models that have provided a way to confront a broad range of problems due to their capabilities to produce embeddings [35]. Tracking with Siamese networks seems to be a promising approach to handle diverse VOT challenges at speeds far beyond real-time [12]. Their primary aim to circumvent the obstacles of pre-trained CNNs by exploiting end-to-end learning for real-time applications [17]. In this section, we describe the fundamental building blocks that are common to Siamese trackers. There are two dominant visual tracking strategies. The first represents models that exploit the classification and updating pipeline. The second strategy focuses on matching-based models [36]. Siamese trackers belong to the latter. This branch of tracking algorithms also belongs to both correlation and non-correlation filter-based ones [33].

The purpose of Siamese trackers is to learn a generic similarity function that can accurately identify whether the two provided image patches belong to the identical object or not. This is the reason where the two input branches come from. Siamese networks are suitable for studying deep neural networks in the context of tracking because they are considered the simplest networks for matching problems [1]. However, viewing these trackers as pure "comparators" would not be adequate. The general similarity function should appropriately handle various visual distortions to the target. We emphasize here that there are no assumptions on similarity. It is entirely described by the training samples. Moreover, since VOT in general does not focus on specific classes of objects, the learned similarity function should also generalize well. Even Tao et al. [28] suitably remarked that they did not attempt to do any offline training of the tracking targets in their Siamese instance search (SINT) tracker because in that case, they would essentially create an object detector instead.

A common pattern in single object tracking is to provide just one exemplar of the tracked object in the initial frame. Learning a visual model from a single example is an ill-posed problem. To obtain a reasonable generalization capability of the created embeddings, a sufficient dataset to learn an invariant representation of generic object features is necessary [37].

The results of [12] demonstrated the expressive power of properly learned similarity function with their Siamese fully convolutional network (SiamFC) architecture. They concluded that training on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset [38] alone was sufficient to attain competitive results against other state-of-the-art trackers on evaluation benchmarks with which their

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

training data had no overlap. The embeddings created by deep learning provide a rich source of features for trackers. These features can handle appearance variations to such an extent that even a front side of an unknown object can often be associated with its backside successfully [23]. As far as the training is concerned, these trackers are trainable end-to-end on given videos using back-propagation, provided that the entire model is differentiable [39]. Authors of [12] were also the first to train a tracker on ILSVRC dataset. Many newer trackers followed their path later on [16], [29], [40], [41]. This dataset is suitable for pre-training, and it seems like it has become a standard. We can say that the SiamFC architecture opened a new branch of deep learning-based trackers [27].

A. SIMILARITY LEARNING

VOT can be modeled as a similarity learning problem [16]. A general similarity learning is addressed in the offline training phase, and then the learned function is evaluated during tracking [12]. Similarity learning is related to learning metric embedding.

As [42] describes, the goal of learning metric embedding is to learn a function (a transformation) $\psi_\theta(x) : \mathbb{R}^S \rightarrow \mathbb{R}^D$, such that $D < S$, that maps semantically similar points from the data manifold in $\mathbb{R}^S$ onto metrically close points in $\mathbb{R}^D$. Analogously, $\psi_\theta(\cdot)$ should map semantically different points in $\mathbb{R}^S$ onto metrically distant points in $\mathbb{R}^D$. Learning such a metric embedding is similar to dimensionality reduction, as it involves mapping a set of high dimensional input points onto a low dimensional manifold. The function $\psi_\theta(\cdot)$ ideally maps similar (a measure of similarity has to be defined) points in the input space to nearby points on the manifold [43] (see Figure 1).

Bertinetto et al. [12] proposed to learn a function $f(z, x)$ that would compare an exemplar image z with a candidate image x of equal dimensions. Later on, they employed strategies that allowed them to relax this constraint on dimensions. Their design decisions incorporated fully convolutional networks. This aspect is important because a great deal of Siamese trackers follow this trend nowadays. The function $f(z, x)$ would return a high score in case of equal objects, and a low score otherwise. Siamese networks apply identical transformation $\varphi(\cdot)$ to both inputs and then combine them using a function $g(\cdot)$, such that $f(z, x) = g(\varphi(z), \varphi(x))$. If we assume the function g to be a distance or similarity metric (e.g., l_2 or cosine), then the function φ can be thought of as an embedding. During inference, the nearest neighbor is used to find the most similar object in the search region [40].

We review trackers that are built upon the idea of similarity learning. Among other things, embedding trained this way can be used to produce feature vectors for classification, one-shot learning tasks [44], clustering [45], face recognition [46], and object re-identification [47].

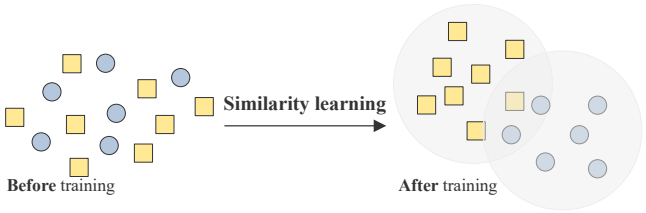

FIGURE 1. A transformation achieved by similarity learning. The goal of the training process is to find a mapping of the objects such that similar objects are mapped closer together in the embedding space while different ones are mapped further away. The criterion for similarity as well as its degree is usually implicitly provided by the designer and the training data itself. The neural networks are capable of discovering this latent structure.

B. SIAMESE ARCHITECTURE

In our paper, we discuss Siamese architectures. The pioneering work in this area of trackers is considered SINT [28]. To be accurate in terminology, and for the sake of completeness, Siamese networks come in the following types: two-stream, two-channel, recurrent, pseudo, and pure (see [1] for more details). Considering the papers we surveyed here, not all of these types have equal occurrence. We show the basic difference of the most important types in Figure 3. Unless stated otherwise, we refer to all these types as "Siamese".

We consider it important to describe the fundamental building blocks of Siamese trackers to enhance understanding. The goal is not to provide an exhaustive comparison of every architectural detail. It can be found for 9 prominent trackers in [1], with 7 of them in common with this paper.

A general pipeline of a majority of Siamese architectures starts with a feature extraction part performed by CNNs. From the standpoint of computer vision, these branches can be thought of as a transformation of visual descriptions of increasing spatial receptive fields. They produce feature maps that are embedded in a measurable space. Later on, the extracted features are merged to assess their similarity (see Figure 4). Once the similarity is computed, then the loss function is evaluated.

Robust visual feature extraction in VOT is of paramount importance. Many architectures exploit pre-trained models on ImageNet dataset [49] to extract features that are then transformed for subsequent similarity learning. Since the visual features for exemplar and search image are often extracted by the same neural network backbone, their weights are usually shared, as in the Generic object tracking using regression networks (GOTURN) [50] framework (see Figure 3). In light of the Siamese networks terminology, we denote $\varphi(z)$ and $\varphi(x)$ as features extracted for the exemplar and search image, respectively. As long as no online model updating is performed, the value of $\varphi(z)$ is usually computed only once during the initialization.

As [12] succinctly described, a function is fully convolutional if it commutes with translation. Specifically, let L_τ denote the translation operator, such that $(L_\tau x)[u] = x[u - \tau]$. Then, a mapping of signals to signals given by the function h is fully convolutional with integer stride k if

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

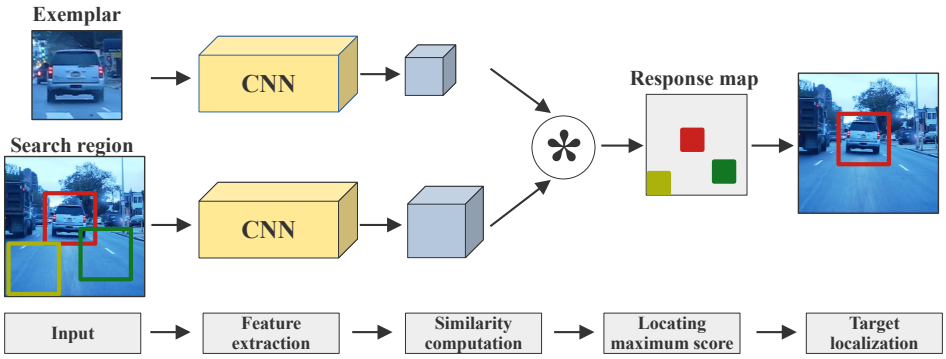

FIGURE 2. An illustration of a fully convolutional Siamese architecture. Typically an exemplar image and a larger search region are fed into two branches of the Siamese network. The extracted features are subsequently merged by cross-correlation operation which produces a response map. Individual cells of the output response map correspond to individual patches from the search region. A location prediction corresponds to the patch with the maximum similarity score. More concretely, assume the exemplar and search images are represented by 3D tensors described by [channels, width, height] of size $[3, 127, 127]$ and $[3, 255, 255]$, respectively. The convolutional backbone (yellow part) produces their corresponding embeddings with sizes $[256, 6, 6]$ and $[256, 22, 22]$, if we assume that the number of feature maps of the last convolutional layer in the backbone is 256. Then, the cross-correlation later (star) uses the $[256, 6, 6]$ tensor as a kernel (weight) and exhaustively convolves this kernel with a larger search region represented by the $[256, 22, 22]$ tensor and produces the response map of size $[1, 17, 17]$. The illustration, as well as the computational example, were inspired by [12].

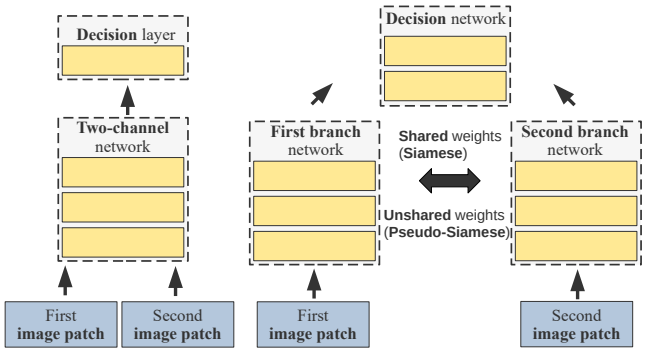

FIGURE 3. Three basic Siamese network architectures: two-channel (left), Siamese and Pseudo-Siamese (right). The Pseudo-Siamese architecture does not use weight-sharing, apart from the basic Siamese architecture (diagram inspired by [48]).

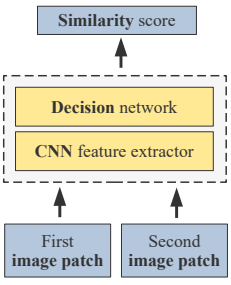

FIGURE 4. An abstract overview of the Siamese architecture. The input consists of two image patches (BBOXes). The core model extracts CNN features and feeds them into the decision network the output of which is the similarity score between the two images (diagram inspired by [48]).

$$h\left(L_{k\tau}x\right) = L_{\tau}h\left(x\right), \tag{1}$$

for any translation τ. This crucial property allows providing a much greater search image to compute the similarity scores across all translated sub-windows in just a single evaluation (see Figure 2). From a mathematical perspective, the formula above represents one of the fundamental traits that stand behind the success of similarity learning applied in object tracking; and that is the translation equivariance [23].

Fully convolutional architectures avoid using padding, even though it is a common practice in CNNs. The reason is that padding violates the property of equation (1) [12], [20]. Another thing to consider is that fully convolutional neural networks also avoid fully connected layers. As a result, they can handle a variable input image size. An efficient solution is to replace fully connected layers with 1×1 convolutions notably propagated in Network In Network [51] model. 1×1 filters were also used in the Inception architecture for dimensionality reduction and at the same time to increase the dimensionality of feature maps [52]. Apart from tracking, fully convolutional architectures can generally be applied to other image processing tasks, e.g., segmentation [22] or detection [21].

Bertinetto et al. [12] introduced the cross-correlation operation. In SiamFC architecture, the cross-correlation was part of a layer that merged the two inputs (exemplar and a search image) to produce a response map. It is also called a score or a correlation map. In this survey, we use these terms interchangeably. This response map (practically visualized as a heat map) represents the scores of the similarity between the exemplar patch and the search region. The idea behind producing this map is to generate a 2D feature map produced by a standard 2D convolution operation in a neural network. The distinction between the cross-correlation and the convolution operation is not important here to convey the essence. Originally, this map contained only one channel, but as we will see later, upcoming works employed multiple channels.

The fully convolutional nature provided a way to compute a similarity score for each translated sub-window of the exemplar image within a larger search region. By exploiting the embedding function $\varphi\left(\cdot\right)$, the produced feature maps can be combined using a cross-correlation layer, thus

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

$$f(z,x) = \varphi(z) \star \varphi(x) + b\mathbb{1}, \tag{2}$$

where $b\mathbb{1}$ represents a bias term which has a value $b \in \mathbb{R}$ in every location, i.e., an offset of the similarity values. The cross-correlation is denoted by $\star$. Given this, the output is not a single score, but a score map defined on a finite grid of positions $\mathcal{P} \subset \mathbb{Z}_+^2$. See Figure 2 for illustration of the process of computing the response map.

C. LOSS FUNCTION

Here we describe the two fundamental loss functions for Siamese trackers. The purpose is to convey the essence of how the evaluation works.

1) Margin contrastive loss

Loss functions used in similarity learning (section III-A) should generate feature representations that are close in the embedding space for positive pairs and far away by at least a given margin for negative pairs. Thus, the contrastive (pairwise) loss [53] for a pair consisting of samples x_i and x_j, their corresponding label $y_{ij} \in \{0, 1\}$ (negative and positive sample, respectively), and margin ϵ is given by

$$\mathcal{L} = \frac{1}{2} y_{ij} D^2 + \frac{1}{2} (1 - y_{ij}) \max\left(0, \epsilon - D^2\right). \tag{3}$$

Usually $D = \|\varphi(x_i) - \varphi(x_j)\|_2$, that is the l_2-norm of two normalized latent representations. This loss was utilized by SINT [28].

2) Logistic loss.

The cross-correlation operation produces a response map that is not a vector but a spatial response map of similarity scores. The logistic loss function from [12] is accommodated to handle this. When training the network on positive and negative pairs, the logistic loss function

$$l(y,s) = \log\left(1 + e^{-ys}\right) \tag{4}$$

is used, where $s \in \mathbb{R}$ is a similarity score for a single exemplar-candidate pair and $y \in \{-1, 1\}$ is the ground-truth label for negative and positive pair, respectively. Thanks to the fully convolutional architecture a larger search region can be used. This produces a map of scores. Let $v : \mathcal{P} \to \mathbb{R}$ be a mapping from a set of positions in the response map to a single similarity score. Then, the loss function is extended by averaging individual losses as

$$\mathcal{L}(\mathbf{y}, v) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} l(\mathbf{y}[p], v[p]), \tag{5}$$

such that each position $p \in \mathcal{P}$ in the score map has a ground-truth label $\mathbf{y}[p] \in \{-1, 1\}$.

The logistic loss was adopted in trackers SiamFC [12], Correlation filter network (CFNet) [54] and Dynamic Siamese network (DSiam) [55]. This loss was a foundation to many upcoming trackers, e.g., Siamese classification and regression networks (SiamCAR) [20] or Foreground information guidance for Siamese visual tracking (FIGSiam) [56], where they extended this loss by adding more terms.

IV. MAIN CHALLENGES OF SIAMESE TRACKERS

Visual features not based on deep learning had undergone the scrutiny of Li et al. [5]. The authors concluded that the existing appearance models (not based on deep learning) were incapable of simultaneously delivering tracking robustness and tracking accuracy. We can say that these issues are still present during times of deep learning. Marvasti-Zadeh et al. [17] analyzed the existing challenges in VOT in general and compiled a thorough discussion about current approaches. Pflugfelder [1] pointed to the need of incorporating additional mechanisms (e.g., filtering or attention) to rudimentary Siamese trackers to improve their performance. Learning a powerful feature representation is vital to achieving a robust Siamese tracker [57]. For example, Bertinetto et al. [12] also closed the evaluation of their SiamFC by saying that they could achieve better results with model updating, BBOX regression, fine-tuning, or memory. In this section, which is the main contribution of this paper, we will describe various traits of the Siamese frameworks, their corresponding issues, and how existing approaches tackle them.

A. BACKBONE SELECTION AND PRE-TRAINING

CNN-based trackers have reaped great success thanks to powerful feature-extraction capabilities. However, most of these trackers exploit backbones originally trained for image classification. As a result, this leads to sensitivity to distractors (more in section IV-C) because CNN models pre-trained for classification tasks emphasize inter-class differences [58]. This may weaken the power of CNNs since there is a significant difference between classifying an object and predicting its location in the image [19]. The objective of the classifier is not coupled to the objective of the tracker [59]. The trackers may then suffer from inconsistency problems caused by task differences. Moreover, pre-trained networks are suboptimal and the performance can be considerably improved by training the backbone network for visual tracking from scratch [17].

He et al. [60] performed an extensive study of the ImageNet pre-training paradigm. They claimed that ImageNet pre-training did yield an observable improvement, nevertheless, for object detection, the improvement was small and scaled poorly with the pre-training dataset size. We consider object detection a task a lot more similar to object tracking than object classification, so we think the same may be true for object tracking. Furthermore, they also showed that ImageNet pre-training brought no benefit in situations when the metric measured the accuracy of predicting the target location. In such cases, random initialization and training from scratch produced evident improvements for BBOX overlaps for high thresholds. So the model reached an improved target localization on the pixel level. Besides, they noticed a faster convergence.

The Siamese region proposal network (SiamRPN) [61] belongs to state-of-the-art trackers. It builds on the idea from [62] concerning region proposal network (RPN). There is also an improved Siamese region proposal network++ (SiamRPN++) version [63] that uses ResNet [64] as a backbone. A known fact is that deep architectures give better tracking performance than shallow ones [20]. Deeper and wider networks produce features that aid in distinguishing the target from its background. However, it is still difficult to have a Siamese architecture that generalizes well, is accurate, and fast while being very deep [65].

A simple glance on the "backbone" column in Table 2 on page 24 indicates a prevalence of AlexNet-like backbone architectures, especially in years 2016-2018. These backbones are relatively shallow since only the first 5 convolutional layers are often used. The reason for this trend is partially due to a preference of successors of SiamFC tracker to use the same backbone for objective comparison. But there is another, more important reason. Remember the embedding matching function defined in equation (2) on page 6. This function inherently implies two restrictions that a Siamese tracker should obey [63]:

- The feature extractor part, as well as the contracting part (cross-correlation) of the network, must follow the *strict translation invariance* property, specifically

$$f\left(z, x\left[\Delta\tau_j\right]\right) = f\left(z, x\right)\left[\Delta\tau_j\right], \tag{6}$$

  where $\Delta\tau_j$ is the translation shift operator for the subwindow. As a consequence, this design choice makes the training and inference significantly more efficient.
- The contracting part poses an intrinsic restriction for *structure symmetry* given by

$$f\left(z, x'\right) = f\left(x', z\right), \tag{7}$$

  which is a natural requirement for similarity learning.

A detailed analysis of Li et al. [63] showed that the two requirements above prevent Siamese trackers from using deep neural networks as their backbones. The main reason is that using padding in convolutional layers as part of deep neural networks breaks the strict translation invariance defined in equation (6). This property only exists in architectures with no padding, e.g., the modified AlexNet architecture that is so common in the Siamese community. To satisfy this restriction, the backbones of Siamese trackers were purposefully designed to be shallow. For instance, if the backbone were replaced by ResNet [64] or MobileNet [66], the padding would be inevitable to make the network deeper, hence, breaking the strict invariance restriction. Directly training the tracker using a deeper network does not yield the expected performance gain. To circumvent this, the very same authors did another analysis in their paper and showed that once they eliminated the center bias, then any off-the-shelf deep neural networks could be adopted as a backbone. Thus, to exploit deep neural networks for feature extraction that involve padding, translation has to be a part of data augmentation. Such sampling strategy effectively alleviates breaking the strict translation invariance property that networks with padding by their very nature do not conform to. This discussion will also continue in section IV-C3, where we will tackle the center bias of Siamese trackers.

This analysis opened up the possibility to use a broader set of backbones, and successive works (e.g., [56]) also adopted the spatial-aware sampling strategy to counterbalance the breaking of strict translation invariance. An analysis by Han et al. [67] also confirmed that using padding in Siamese backbones had a huge negative impact on tracking performance. They also followed the recommendations above and adopted the same strategies to evade the problem of breaking the strict translation invariance caused by the use of padding. To demonstrate the effectiveness, they proposed Fully convolutional anchor-free Siamese network (FCAF) tracker. Their additional contribution was to avoid anchors in the RPN. This approach utilized pixel-level classification and regression, too.

The reasoning above concerned only the strict translation invariance requirement. Most of the trackers, by their very nature, obey the second requirement regarding structure symmetry (equation (7)). But the SiamRPN framework broke the structure symmetry by design, too. This was another motive for the authors of the SiamRPN++ framework to perform their theoretical as well as experimental analysis. They remarked that the bounding box prediction and anchor-based classification were both asymmetrical. Moreover, convolutional layers in the exemplar and search branch were not shared, and that also produced asymmetrical features, making the training less stable (more in section IV-B). We may digress a little, but one interesting fact is that the asymmetrical structure of RPN-based trackers makes them more vulnerable to adversarial attacks. They can be effectively misled to classify the target as background by directly attacking their classification branch in all anchors [68].

Authors of [63] claim they also benefited from the ResNet architecture by a layer-wise feature aggregation for the cross-correlation operation. This enabled the tracker to predict the similarity map based on features learned at multiple different levels. Speaking of feature aggregation across multiple levels, there is another extension of the SiamRPN architecture developed by Rao et al. [69] under the name Feature pyramid Siamese region proposal network (FPSiamRPN). As the name suggests, the main contribution of this work is the adoption of feature pyramid network (FPN). The core idea of the FPN is to harness features across multiple levels. Let us provide an example. Assume a usual pipeline of a convolutional neural network. At every level, this network decreases the spatial resolution of feature maps, increases the effective receptive field as well as the number of channels. Low-level features are captured at the beginning of the network, whilst high-level features at the end. FPN is a general extension to CNNs for concatenating features across the bottom-up and top-down direction. The bottom-up direction is a standard

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

processing pipeline described above. The top-down direction consists of up-scaling and adjusting the number of channels. Later on, feature maps with matching dimensions (those on the same level) can be added together. Each concatenation then serves as a basis for a unique prediction. This modification also brought an outstanding performance to object tracking.

A substantial performance improvement can be attributed to the different backbone itself. The work of Li et al. [70] also supported the claim that just a different backbone may be responsible for the improvement of a Siamese tracker. They showed it with the performance of their Siamese with VGG network (SiamVGG) using a customized VGG-16 [71]. They remarked that the often-used AlexNet backbone had limited feature-extraction capabilities. Nevertheless, it is important to emphasize that not all networks are suitable for the Siamese structure because of the two aforementioned requirements.

Using pre-trained backbones seems to be a general issue even with trackers that are not Siamese. For instance, Wang et. al [13] also commented on their results for Fully convolutional neural network-based tracker (FCNT) that the primary failure cases were related to handling situations with low resolution. They conjectured that a possible reason was the VGG [71] backbone trained on ImageNet dataset [49] using high-resolution training images.

Besides all this, there is a risk of dataset bias itself [72]. The data bias of ImageNet is different from that of the data observed during tracking [37].

B. RESPONSE MAP DESIGN

1) Number of channels

One of the drawbacks of the basic cross-correlation layer from [12] is that it only generates a single-channel response map, which lacks relevant features [20]. SiamCAR tracker [20] exploited depth-wise correlation layer to generate a multi-channel response map. Building on top of the established principle of creating a response map with a single channel in Figure 2, a multi-channel response map contains multiple single-channel response maps stacked along the channel dimension. It provides more information (produces richer embeddings) and opens up the possibility to dynamically choose which channels are important in certain situations. This idea is at the core of attention mechanisms which we will discuss in section IV-E2.

Li et al. [63] commented that different feature channels extracted different semantic information. In SiamRPN [61], the feature extraction was extended by introducing another convolutional layer to scale the channels. Due to this extension, the cross-correlation operation could embed even more information, such as anchors. An undesirable consequence was a parameter imbalance that made the training more difficult. Li et al. [63] analyzed the Siamese network structure and found out that its two network branches were highly imbalanced in terms of the number of parameters. The exemplar and the search image branches passed through two non-shared convolutional layers. Therefore, this follow-up work brought a lightweight, depth-wise cross-correlation layer, instead. This modification not only vastly reduced the number of parameters, but it stabilized the training, too. The cross-correlation can also be extended by multi-layer fusion described in DSiam paper [55]. They computed the final scores of the response map as an element-wise weighted sum of scores obtained from different depths of the feature hierarchy.

Siamese network with segmentation mask (SiamMask) [2] architecture also exploited depth-wise cross-correlation [73] to encode richer information about the target object. Moreover, this tracker relied on multiple response maps individually, dubbed as a response of a candidate window. It represented a similarity between the exemplar z and n-th candidate window in the search image x. This architecture also exploited the availability of more information in a multi-channel response map to generate a binary segmentation mask of the target during the tracking process. It helped to predict a rotated BBOX, not just axis-aligned. A follow-up work of Siamese network with segmentation mask and ellipse fitting (SiamMask-E) [74] improved the BBOX prediction by ellipse fitting. However, VOT datasets usually only consist of axis-aligned BBOXes, so they had to employ special datasets even for the tracking part, not just the segmentation.

2) Spatial granularity

Li et al. [29] pointed out that the size of the response map in the SiamFC architecture was relatively small (just 17×17). As a result, it is not suitable for precise positioning. To this end, their Siamese network with re-detection mechanism (SiamRM) utilized fewer strides (4 in total, instead of 8) to obtain a larger feature map. Since they exploited re-detection mechanisms, a more granular feature map was beneficial to pinpoint the object location with higher precision. To maintain real-time speed, they reduced the number of channels in the last layer from 128 to 32. Likewise, Tao et. al [28] commented that the tracking problem is practically a localization task, thus being susceptive to rough discretization. To achieve precise localization, they employed very few max-pooling layers in their SINT tracker. They also argued that the max pooling operation served the purpose of suppressing local deformations that were pertinent for classification tasks. However, in tracking, the object changes its appearance over time, thus the tracker should follow minor appearance changes between frames.

3) Score values

For completeness, we will also mention the probability map. One of the works in tracking where the authors considered generating a probability map instead of producing a class label spatial map was [37]. The probability map, as opposed to the response map we mostly refer to, is bounded between zero and one, such that values of particular pixels close to one indicate the object's presence. This approach was employed in Two-flow convolutional neural network

(YCNN) [19] architecture. However, the majority of the Siamese trackers nowadays use response maps that are unbounded. It was argued that the optimization is then insufficiently bounded [1]. To address this, Discriminant correlation filter network (DCFNet) [75] tracker used weight decay as a regularization in their l_2 loss function. But the use of cross-entropy loss (as in SiamFC [12] or CFNet [54]) resolved this issue, too. The YCNN tracker also used just the l_2 loss. Nevertheless, they adopted a probability map rather than a response map, and this made the optimization well constrained by itself [1].

C. SIMILAR INTERFERENCE

1) Distractor awareness

The term similar interference describes a situation where a tracker faces the problem of discerning between multiple similar-looking objects. The tracking process is thus interfered with by similarity. Siamese networks have drawn great attention due to their balanced accuracy and speed. On the other hand, these networks extract features that are useful for the discrimination of foreground from a non-semantic background. Semantic background, a.k.a. distractors, pose a risk to the robustness of Siamese trackers [40].

Siamese tracking is essentially a template matching problem, and most template-based trackers generally fail to track an object that undergoes significant changes in appearance [29]. This flaw is primarily caused by a misalignment between the general domain and the specific target domain [40]. Besides, Siamese architectures usually process one local neighborhood at a time. This property makes them non-robust to abrupt appearance changes of the object [65]. Li et al. [70] also highlighted weak discrimination capability as a major drawback of the current Siamese trackers. Historically speaking, SINT [28] is considered the first Siamese tracker. In light of the template matching process, they approached tracking as a verification problem. Authors commented in their evaluation that when similar objects appeared at the same time, the tracker sometimes jumped from the correct target to the distractor.

Even the region proposal-based SiamRPN [61] tracker, despite multiple enhancements, still had difficulties with distractors for the tracked object [20]. A follow-up work of [40] under the named Distractor-aware Siamese region proposal network (DaSiamRPN) increased the hard negative mining in the training phase. This data enhancement step led to improvement in discrimination and produced more robust performance. Additionally, this tracker had a special distractor-aware module. This module adopted the non-maximum suppression (NMS) [76] algorithm to select potential distractors in each frame. This tracker with distractor-awareness could achieve adaptation of the existing general similarity metric to a domain-specific similarity metric. A new distractor-aware objective re-ranked the candidates for distractors according to similarity with the exemplar. These techniques practically built an online trained classifier. Last but not least, this tracker was capable of long-term visual tracking thanks to its ability to reason not only about similar-looking objects but also whether the target was present or not. Even though long-term tracking bears the greatest transfer to real-world applications, very few trackers have been proposed for this task to date [17].

Li et al. [29] proposed the SiamRM architecture. The goal was to boost the SiamFC in complex scenes with fast motion and the presence of distractors. In situations when the response map contained multiple peaks, a special re-detection mechanism based on SINT [16] is executed. Moreover, they employed GMM to dynamically update the template instead of using a fixed template from the first frame. But, they did not update the template in every frame or a fixed interval. Instead, they updated only templates with high confidence to enhance the quality. Template updating provides benefits to model adaptation, but online tracking may become very inefficient (discussed more in section IV-E). Still, it does not solve the tracking drift problem caused by similar interference completely [20].

Voigtlaender et al. [77] pushed the idea of re-detection even further. Their Siamese Faster R-CNN re-detector (SiamR-CNN) framework unleashed the full potential of two-stage object detection, specifically of Faster R-CNN [62] architecture. This tracker was capable of object re-detection after long-term occlusion. However, a tracking-by-detection approach is inherently required to manage the detections that could potentially belong to distractors. To this end, the authors developed a novel algorithm based on dynamic programming to take the advantage of re-detections of the exemplar from previous frames to model the full history of the tracked object. Besides, they also proposed a novel hard example mining strategy to promote better robustness to distractors.

2) Feature extraction

This section elaborates on a prevailing trend in the exploitation of features from multiple layers of the backbone. Most Siamese trackers nowadays learn the high-level appearance features of the entire object. This trait increases their tendencies to suffer from drift problems caused primarily by non-rigid appearance deformation or partial occlusion [57]. The drift problem is also a target of researchers because once the tracker loses track of the object, it is difficult to recover from such a state without additional coping mechanisms. We consider feature extraction pertinent when dealing with distractors. We think that robust feature extraction is the minimum requirement for a tracker to reliably delineate the boundary between a foreground and a background. Besides, powerful features may serve for discerning between similar-looking objects, too. We thus believe that the exploitation of features extracted from multiple levels is becoming more of a rule than an exception.

The work of Abdelpakey et al. [65] extensively relied on the combination of features from multiple levels. Motivated by densely connected CNNs [78] as well as attention mechanism from [79], there is the Densely-Siamese network

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

with self-attention model (DensSiam) [65]. This framework tackled the problematic nature of too deep Siamese networks together with their focus on the local neighborhood. This tracker employed the concept of dense blocks. A dense block consists of multiple fully connected layers where each layer is connected to all subsequent layers in a feed-forward fashion. It is simple in construction yet effective in performance. The architecture consisted of two identical branches, except that the branch responsible for processing the search image (dubbed as "target branch" in the article) contained an additional attention module at the end. Each branch started with a convolutional block and then continued with several alternating dense blocks and transition blocks to align the tensor dimensions. The standard cross-correlation layer from SiamFC fused the two outputs to produce the response map to learn the similarity function. DensSiam enhanced the generalization ability by producing a response map based on non-local features (the ones that capture the object neighborhood, too) that were robust to appearance changes. This model allowed both low-level and high-level features to flow through the network while avoiding vanishing gradients (thanks to dense connections) which improved the generalization capability.

Another recent work from Luo et al. [80] dubbed as Siamese network with information fusion and rectangular window filtering (SiamFF) employed very effective and intuitive multi-level feature fusion in their backbone with 5 convolutional feature levels, following the "standard" AlexNet-like pattern [12]. But still, they modified the architecture by removing padding in layers and altering the number of network channels. They fused features from levels 2 with 4 for both exemplar and search images and produced a temporary response map. Analogically, they fused features from levels 3 with 5 and obtained another response map. Subsequently, after adjusting tensor shapes, these response maps were merged to produce the final response map. Their other contribution, a score map filtering strategy, will be discussed in section IV-C3.

The SiamCAR [20] architecture used a modified ResNet-50 [64] as a backbone. They combined multiple features extracted from the last three residual blocks and fed them into an improved depth-wise cross-correlation (section IV-B). They attributed the improved recognition and discrimination of their tracker to this design decision. A very similar architecture to SiamCAR that we introduced in section IV-A is FCAF [67]. Authors adopted feature fusion from a ResNet backbone, too. The introduced trivial, yet effective feature fusion module consisted of aligning the extracted feature tensor dimensions using 3×3 convolutions and then performing element-wise addition. Their analysis and comments on the use of low-level as well as high-level features conform with the existing trend we have encountered.

Liang et al. [57], who developed the Local semantic Siamese network (LSSiam), showed that learning not only high-level appearance features but also local semantic features (more robust to non-rigid object deformations) that contain more fine-grained information about the object are essential for attacking the drift problem. Their approach consisted of forcing the model to pay attention to fine-grained details about object appearance by adopting a classification branch to learn semantic features as part of the offline training. Additionally, they extended the classical Siamese framework by a generalized focal logistic loss [81] to mine hard negative samples. To preserve high computational speed during inference, the newly introduced classification branch was removed and replaced by an effective template updating strategy. They argued that even successful approaches like SiamFC [12], DSiam [55], Semantic-appearance Siamese network (SA-Siam) [16], and Residual attentional Siamese network (RASNet) [41] ignored the local or semantic information of targets during offline training and focused on global features instead.

Besides the standard VOT, there is also thermal infrared (TIR) object tracking, for which powerful feature selection is crucial. TIR tracking has to deal with similar objects in the infrared spectrum that may be otherwise unambiguous in RGB. Thus, similarity interference is just reinforced. For enhanced feature fusion, Li et al. [59] proposed their Hierarchical spatial-aware Siamese convolutional neural network (HSSNet) for TIR object tracking. Their method is based on coalescing multiple hierarchical convolutional layers in conjunction with a spatial-aware network. The incentive was to combine spatial information of the shallow layers for precise object localization with deep semantic features to distinguish between objects. The feature coalescing exploited max-pooling layers for tensor alignment and batch normalization to balance the influence of individual feature maps. However, the concatenated features were not robust to spatial variation (rotation, translation, and scaling). To address this problem, they employed spatial transformer network (STN) to learn a 6 degrees of freedom (DoF) affine transformation that was then applied to the feature map.

Another effective strategy is to employ entire subnetworks to process low-level and high-level features before fusion. Liu et al. [82] proposed Multi-level similarity Siamese network (MLSSNet) tracker to better handle distractors as part of the TIR tracking task. In their work, the low-level and high-level features extracted from the backbone were treated by structural and semantic correlation similarity networks, respectively. Low-level features were first processed by two convolutional layers followed by two deconvolutional layers to map the feature location back to the original image. The final layer of sigmoid nonlinearity produced a 2D weight map representing the importance of individual local structures. At the same time, semantic features were squeezed into two 1D vectors using global average and max-pooling layers. To establish a relationship between the two produced vectors, they used fully connected layers followed by a sum operation. Outputs of both of these two networks were processed by a correlation filter in isolation to produce two response maps. The resulting response maps were fed into the relative entropy-based adaptive ensemble network to obtain optimal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

similarity simultaneously encompassing structural and semantic features. The goal of this module was to generate the final response map which had a minimum distance between the structural and semantic similarities, practically making a compromise.

Following on the TIR tracking task, Liu et al. [83] developed a Multi-task matching network (MMNet) tracker, in which they processed backbone features in three different branches. Their complex mechanism involved a classification branch to obtain TIR-specific discriminative features. The classification was supposed to drive the model into learning to discern between different objects. Then, these features were fed into the discriminative matching branch, which exploited the CFNet [54] tracker in a role of a general matching architecture. They showed how an existing Siamese tracker can be utilized within a larger domain-specific tracking pipeline. Their solution for producing robust features also encompassed a third branch focused on fine-grained feature matching.

The fully convolutional approach of FCNT [13] also exploited various useful features from the backbone. These features were fed into general and specific network branches. The general branch aimed to capture the category information. This branch relied on feature maps placed further in the network model, i.e., the high-level features. Conversely, the purpose of the specific branch was foreground/background discrimination. Here, the low-level feature maps produced closer to the beginning of the model were important. These two branches were then hooked on the chosen features to process them further. Later on, their output was combined and served for score map regression for target localization. They also adopted multiple online update strategies. We consider this work important since they comprehensively discuss feature properties of CNNs under the viewpoint of visual tracking.

3) Object location

The assumption is that the highest value in the response map corresponds to the location of the object. But this is not always true, especially in scenes involving similar interference (distractors). In such a case, the object location may correspond to the non-maximum response value. The solution with cosine window from SiamFC [12] effectively suppressed the responses at the boundaries. The original SiamFC as well as some other successors such as SiamRPN [61] used cosine window in endeavor to suppress distractors (see Figure 5). However, when the object moved too fast, the track was often lost [29]. This problem is common to algorithms based on correlation filters. The use of the cosine window contributes to their weak ability to distinguish distant objects from the center. Remember that thanks to the fully convolutional property we can center and crop the search window to the previous object position. This means that, ideally, the peak value in the response map should emerge in the center, regardless of where the object is present in the image. The cosine window builds on this assumption and weights the

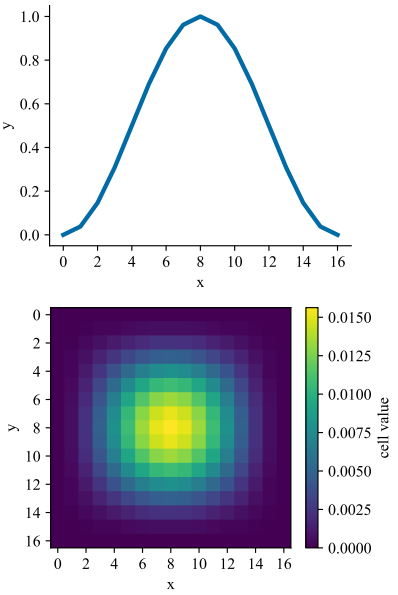

FIGURE 5. A visualization of 1D (top) and 2D (bottom) cosine window if we assume a response map of size 17×17. The 2D version can be simply created as an outer product of the two vectors corresponding to the 1D version. This mask is very similar to Gaussian probability distribution and conveys the idea of putting the highest weight to the center with nonlinear, even reduction when moving away from the center. However, in this case, the cosine function is used instead.

response in the center a lot more than on the edge. Simply put, the closer the peak value is to the center of the response map, the higher the weight is. This strategy improves the stability of the algorithm, but it is risky. During inference, the crop is centered on the last predicted location, which is assumed to be the tracked object. Thus, the target will almost always be affected negatively by this penalty due to its movement [68]. A small search window may mitigate similar interference, but on the contrary, fast-moving objects may become suppressed. Hence, the track may easily drift to the background. Besides, it was demonstrated that thanks to the use of a cosine window, an adversarial attack on Siamese trackers can be easily performed [68]. But simply removing the cosine window would not be sufficient, because then the tracker would not be robust to distractors or sharp noises. With this in mind, we suggest paying extra attention to accurate object centering and location prediction. All in all, this technique, in its original form, works reasonably well, as demonstrated by a plethora of successive works, e.g., [57], [61], [23].

In light of the possibly "faulty" weighting of the regions by the cosine window, some works employed additional motion estimation strategies to better delineate the region where to search for the object. To this end, Luo et al. [80] proposed their SiamFF architecture. They exploited the continuity and stationarity of the movement of objects in reality and developed a score map filtering strategy. Their approach consisted of using a rectangle representing a region where the object could be present based on inter-frame information, hence the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

"rectangular filtering" in the tracker name. This bypassed the limitation of a spherical cosine window and adjusted the "shape" (rectangle instead of a circle) of the response map suppression based on the current context.

The re-detection mechanism from SiamRM [29] brought an improvement. SiamRM removed the cosine window completely and simply re-detected the object from scratch if there were multiple peaks in the response map. They executed the re-detection when the value of the secondary peak in the response map was 0.75 times the primary peak. The authors also devised a way to measure the quality of the response map using average peak-to-correlation energy. This measure could indicate the presence of occlusion or similar interference.

Another important problem to mention is the center bias in Siamese trackers. Bertinetto et al. [12] remarked that the fully convolutional property made the model invariant to translation, hence there was no need to do translation during the data augmentation (they only applied a negligible random translation by 4 pixels). Moreover, they believed that it was effective to consider search images centered on the target since the sub-windows with the greatest influence on the performance were the ones adjacent to the current target. Nevertheless, the authors of SiamRPN++ [63] tracker observed a center bias in Siamese frameworks. They performed an extensive analysis of this phenomenon and developed appropriate measures to address this issue. More specifically, they studied three different scenarios where the object was uniformly translated as part of data augmentation by either 0, 16, or 32 pixels. After convergence, they aggregated the resulting response maps and normalized them to easily represent probability distributions using heatmaps. The subsequent analysis showed that in the case of the 0-shift scenario, the probability values at the edges of the heatmaps degraded to zero. This indicated that the model settled for this biased solution regardless of the appearance of the test target during the inference phase. The other two scenarios demonstrated a tendency of the model to avoid this trivial solution. Their quantitative results indicated that the resulting heatmaps produced by a shift of 32 pixels were closer to the ground-truth location distribution of test objects. To further avoid placing a strong center bias on objects, they devised a spatial aware sampling strategy that essentially sampled the target from the search image by a uniform distribution. They attributed the success of their SiamRPN++ framework to randomly shifting the object location in the training phase, among other things. Another argument against the inherent translation invariance came from Pflugfelder [1] who remarked that this property impedes the exploitation of spatiotemporal information. Trackers like these may be more prone to drifting to the background or different objects, which is an existing and serious issue.

Contextual information seems to be another thing to consider. The authors of SiamFC [12] employed another design choice to incorporate additional context information to the crop of the image region. The context was introduced into the model as a simple enlargement of the cropped region for the exemplar (smaller patch, e.g., 127×127) and the search instance (larger patch, e.g., 255×255). The fully convolutional property of their network allowed seamless use of input with variable dimensions. These regions were computed such that the object of interest was not distorted and fit the entire view. Contextual information was also explicitly processed using recurrent neural networks (RNNs) in the work of [84]. Their recurrent Siamese architecture enhanced the similarity matching by leveraging contextual information. The RNN was adopted to memorize long-range contextual dependencies of the tracked object to learn the self-structure information. They implemented a 4-directional RNN to compute the context features that described the tracked object locally as well as globally. As we can see, joining both local and global features has appeared multiple times in our survey.

The context itself may pollute the response map if similar interference is present. In such a case, Li et al. [56] proposed a foreground information guidance module for their FIGSiam tracker. The core idea was based on a padding image (derived from the search image) that contained the target object but the remaining context (outside the BBOX region) was filled with a mean color computed independently for all three channels. This image thus served as a "guidance" for the network to focus on a "proper" foreground. A result of the forward pass through the guidance module was then incorporated into the loss function using a convex combination with the standard logistic loss function for Siamese trackers. Authors visualized response maps as heatmaps and showed that the response values for similar objects were suppressed.

4) Object occlusion

Even though object occlusion deserves a section on its own, we do think that it is closely related to similar interference. Once the object becomes occluded or goes out of view completely, there is an increased risk for the tracker to drift to the background, whether semantic or not.

One obvious approach to simulate occlusion is to cover a specific part of the object region in the image. For example, Li et al. [56] added occlusion as part of data augmentation for training their FIGSiam architecture. Let w and h be the width and height of the ground-truth BBOX denoting the location of the target object. Then, they generated a black rectangle of size $w/2 \times h/2$ and partially covered the object region by randomly selecting one from 11 different but fixed positions. Besides this, they also included rotation and shear transformations to further expand the possibilities of the dataset.

The problem of occlusion is prevailing in all Siamese architectures. Gupta et al. [85] proposed to simulate the effect of occlusion on the level of embeddings (latent space representation). They employed structured dropouts and showed that this extension could be incorporated into existing architectures such as SiamFC [12] and SiamRPN++ [63] and could boost their performance by 3%. Unlike the common forms of dropouts, the structured dropouts were intended to mimic the effect of occlusion in the latent space. Since

the occlusion can present itself in an unlimited number of variations, the authors of [85] argued that their proposal was the only known feasible approach for occlusion handling.

The robust performance of the SiamRPN architecture inspired multiple subsequent works. One of them was proposed by Wu et al. [86]. The authors developed a modified SiamRPN architecture with an anti-occlusion mechanism based on Kalman filter [87] dubbed as Anti-occlusion Siamese region proposal network (AO-SiamRPN). Their incentive was that SiamRPN treated tracking as a local one-shot detection task, so it could not adequately handle occlusion, fast motion, or out-of-view situations. The decision was to exploit the spatio-temporal information about the object to address the mentioned problems. The processing pipeline first extracted object features through a deep CNN and then used an adaptive Kalman filter to predict the target trajectory. This introduced an online updating into the framework. In addition to this, a new hard example discrimination method (HEDM) was proposed to estimate whether the occlusion occurred and if it did, then how serious it was. Such information was also crucial for the Kalman filter updating phase. The Kalman Filter was adopted for capturing the object motion trajectory to make full use of the temporal information to cope with occlusion. The introduced HEDM was employed to supervise the credibility of the Kalman filter. This module not only judged the severity of occlusion it also made the Kalman filtering mechanism adaptive and more robust than the original formulation. As a result, this framework had a different search mechanism from the traditional Siamese trackers. Instead of using the center of the predicted object position in the previous frame, this method utilized the object position predicted by the adaptive Kalman filter for cropping.

D. SPATIAL OBJECT TRANSFORMATIONS

1) Multiscale inference

A single response/similarity map does not contain enough spatial information [20]. A common approach to deal with this is to run matching on multiple scales to determine the object scale variation. For example, SiamFC [12] ran the search at 5 different scales, $1.035^{\{-2,-1,0,1,2\}}$, and then linearly interpolated the object scale with a factor of 0.35 for smooth transition. The authors also introduced a version that searched only 3 different scales. It resulted in considerably higher speed (from 58 to 86 frames per second (FPS)) together with slightly higher accuracy (from 0.524 to 0.534). It seems that there is a reasonable number of scales to search for before the returns start to diminish, but we are not aware of any consensus. SINT [16] used only 3 different scales for an acceptable balance between speed and performance. Nevertheless, this approach still delivered an additional computational burden regardless of its speed. The current trend seems to diverge away from the multi-scale search.

The SiamRPN [61] avoided multi-scale search by use of region proposals, that introduced anchors and a lot of additional hyper-parameters. For this tracker, hyper-parameter tuning was crucial for successful tracking. On the other hand, the SiamCAR [20] architecture is known for its simplicity. While avoiding multiscale search, anchors, and region proposals, it still delivered a state-of-the-art performance with very few hyper-parameters. Their contribution was the formulation of tracking as a regression and classification task. Another trait of their tracker was that the BBOX regression and object classification were performed on the pixel level. Their paper also covered the notion of "centerness", which, on the pixel level, indicated how far away the current pixel was from the object center. The model was trained in a supervised fashion to estimate this value. This quantity was then exploited to improve the BBOX prediction based on the assumption that predictions closest to the object center were the most accurate. Among other things, they also showed that a multi-channel response map produced satisfactory features for both mentioned tasks. Speaking of predictions on the level of pixels, even [37] anticipated in 2015 that pixel-wise approach could help with irregular object shapes. We think this supports the claim that pixel-wise approaches deliver precise object localization, too.

Another very similar architecture to SiamCAR is Siamese box adaptive network (SiamBAN) proposed by Chen et al. [88]. They also treated tracking as a simultaneous classification and regression task and avoided anchor-boxes completely. The motivation behind their work was to exploit the expressive power of fully convolutional networks to avoid the heuristic configurations of target scales and aspect ratios. Their design was easy to use without excessive hyperparameter selection before training, as is the case in RPN-based trackers. The developed box adaptive head consisted of classification and regression modules. Each of these modules received a fusion of features belonging to exemplar and search branches. Due to different tensor shapes, they make predictions at different levels of the backbone. That is one prediction for the i-th convolution layer from search and exemplar branch, another for $(i+1)$-th, and so on. Subsequently, these partial predictions coalesced into the final response. One notable approach in their solution was the use of dilated (atrous) convolutions [89].

2) Transformation equivariance

Real-life scenarios often demonstrate how objects undergo different transformations besides pure translation, for example, rotation and scaling. Sosnovik et al. [23] remarked that unless the model is equipped with an explicit mechanism to cope with such visual distortions, then the similarity could be severely degraded. To this end, there are emerging works that deal with models equivariant to a specific operation, e.g., scaling or rotation.

Regarding the change in scale, Sosnovik et al. [23] proposed an extension to a standard Siamese architecture that learned to handle scale variations offline. They augmented the standard SiamFC [12] architecture and dubbed it as Scale-equivariant Siamese fully-convolutional network (SE-SiamFC). Their contribution was based on scale-equivariant

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

steerable networks [90]. The tracker was capable of capturing natural variations in the target scale a priori. Also, they developed a theory by use of which even already existing trackers could be made scale-equivariant. The proposed recipe briefly goes as follows: 1.) perform a domain-specific estimation of a possible change in object size; 2.) derive the necessary scale-related parameters; 3.) replace all convolutional layers with scale-convolutional layers; 4.) optionally include scale-pooling to capture additional inter-scale correlations between all scales; 5.) replace the cross-correlation operation with a non-parametric scale-convolution.

There is also a counterpart work aimed at rotation equivariance. In the paper by Gupta et al. [91], they demonstrated that the performance of existing trackers was severely affected in presence of rotation instances. To circumvent this obstacle, they proposed a Rotation-equivariant Siamese network (RE-SiamNet). This architecture, similar to SE-SiamFC, exploited steerable filters. They adopted group-equivariant convolutional layers. Such trackers were capable of estimating the change in orientation of the object as well as 2D pose estimation as a by-product. This information could thus be used to establish motion constraints. They showed the efficacy of this approach by outperforming other trackers on custom-curated datasets involving excessive rotation variation. However, they also included evaluations on standard benchmarks and showed a 2% decrease in performance. The reason for this minor drop could be caused that the trackers used a fewer number of channels for the same number of parameters as the original counterparts, thereby exhibited slightly lower discriminative power. Authors experimented with extending the SiamFC [12] and SiamRPN++ [63] architectures.

E. ONLINE MODEL UPDATING

1) Online adaptation

Model updating is often accompanied by a dilemma. On one hand, if the tracker does not update frequently enough, it may not catch the appearance changes appropriately. On the other hand, if the tracker updates the internal object representation too frequently, the risk of encountering the problem of drifting either to the background or a similar-looking object increases [37]. Nowadays, the two most common ways to achieve template updating are either linear interpolation or multi-template updating [92]. As for Siamese trackers, the constant template strategy (i.e., the template is initialized in the first frame and never changed) that is used in this type of trackers exacerbates their ability to adapt to the drastic appearance changes [56]. Trackers that exploit classification and online model updating perform among the best in terms of accuracy, but they are the slowest ones (even 1-2 FPS). Conversely, trackers based on matching (e.g., Siamese) are the fastest, but not the most accurate [36]. Their accuracy-to-speed ratio is probably the best so far. But this claim is not easy to generalize. Besides the comments above, Kristan et al. [36] also introduced their DSiam architecture. This tracker enabled effective online appearance adaptation as well as background suppression. They extended the original SiamFC architecture by further processing the output feature maps from the two branches. Feature maps belonging to the exemplar image were fed into a target appearance variation transformation. Conversely, feature maps of the search image were fed into a background suppression transformation. Only then the cross-correlation operation was executed. These transformations were learned using regularized linear regression. Appearance variation was modeled by regressing an affine transformation between the initial and the current patch. The goal was to find an affine transformation that as closely as possible transformed the current object appearance to the initial appearance. Background suppression was achieved by further regressing the affine transformation to suppress features in the current search region that did not belong to the target patch. In other words, they aimed at filtering features that interfered with the identification of the tracked object by explicitly teaching the model to suppress features that belonged to the background.

Among other things, occlusion poses a major challenge to VOT in general. As long as the tracker does not employ a robust template updating strategy, the occluder may be easily mistaken for the occluded (target) object. So, either the updating mechanism is accurate, or it should rather be avoided. As a result, the tracker may drift to the background [29]. Thus, when it comes to using templates, often more sophisticated strategies are necessary. Siamese trackers extract an appearance template from the initial frame and then use it to localize the target in future frames. Template updating is not often used in Siamese trackers. A simple approach would be to linearly combine the current template with the accumulated templates from the previous frames. But this strategy results in an exponential decay of information over time [93]. Most of the time, the current Siamese tracking methods use the target in the first frame as a template during the whole tracking period. This leads to failures caused by target deformation.

To this end, Xu et al. [94] proposed a new template updating method. Specifically, they based their adaptive template updating module on two different networks, namely a neural contour-detection network, and a target-detection network. The purpose for the introduction of the contour-detection network was to exploit a contour-based proposal template initialized in the first frame instead of a fixed BBOX. As far as the target-detection network was concerned, they adopted a single-stage YOLOv3-like [95] approach with dilated convolutions to expand the receptive fields for more granular object detection. The developed template updating strategy was governed by the maximum value in the response map. If the highest response value fell below a specific threshold, then either the object was deformed or occluded. Their strategy was supposed to differentiate between these two scenarios and decide whether to update the template or not. In case of occlusion, the template should not be updated to prevent pollution. This decision was made based on the results of the contour-detection and target-detection networks. The contour-detection network provided a shape-adaptive

template approach that could handle partial occlusion very well. In such a case, the two mentioned networks should be consistent in their predictions. On the other hand, extreme occlusion or even out-of-view situations caused these networks to not get consistent results most of the time. To increase the robustness of their tracker, another part of their approach was to still remember the initial template as a backup.

Zhang et al. [93] proposed to replace the handcrafted, deterministic update strategies. They used an entire neural network dedicated to learning the optimal template for the next frame given the current and historical observations of the tracked object. The template updating based on GMM in SiamRM together with re-detection mechanism could handle long-term object tracking in presence of occlusion [29], too. This tracker was capable of identifying that the object was not present or that there was some distractor present (as discussed in section IV-C). Once that happened, they adopted the SINT [28] tracker to perform re-detection. Speaking of the SINT framework, this tracker, as reported by the paper, also allowed for an accurate target re-identification even after it was absent for a complete shot. The authors achieved this by utilizing a window sampling over the entire image using edge boxes [96].

A similar approach was adopted by Li et al. [56] in the template updating module as part of their FIGSiam tracker. They decided to combine multiple templates exponentially. This is a common practice that we have observed in multiple works. They used a pool with a capacity for n templates. Each template had a corresponding embedding associated with it. Its similarity score reflected the mean similarity computed as a dot product between the embedding vectors of the current and previous templates. When a new frame emerged (for which the template embedding vector was produced), then there were two situations. In the first scenario, when the pool was not full, and if the similarity score for the current template was above a certain threshold, then this template was added to the pool. Otherwise, it was discarded. In the second scenario, when the pool was full, the similarity score was computed among all the templates stored in the pool and the new template replaced the one with the lowest similarity score. The embedding vector itself was derived from the feature tensor that is usually used as a kernel for the cross-correlation operation in the majority of the works discussed so far. Specifically, if the extracted features had dimensions of $8 \times 8 \times 256$, then they applied global average pooling to produce a vector of size $1 \times 1 \times 256$. In the end, this vector was l_2-normalized.

2) Attention

Offline training offers a reasonable balance between tracking accuracy and speed. However, it is still difficult to adapt a model trained offline to a target tracked online. Online model updating is usually avoided as there is a tendency of deep feature extractors to overfit the target, besides the additional computational overhead [41], [50].

To tackle this, a tracker developed by [41] under the name RASNet reformulated the correlation filter in the Siamese framework and introduced attention mechanisms [79] to adapt the model without online updating. They developed three diverse attention approaches. General attention that was honed by the dataset during the offline training phase; residual attention to aid in adapting an offline model to online tracking (reaping the benefits of offline training and live tracking); and channel attention that reflected the channel-wise quality of features (to improve feature selection). The use of residual learning helped with adaptive representation. The cross-correlation layer was enhanced by a weighing mechanism, thus the problem was reformulated from a regression perspective. The reason was that not all features contributed equally. This work did not employ the standard cross-correlation map that produced a single-channel response map (see Figure 2). Instead, they built on top of the extended response map containing multiple channels, but the underlying principle remained the same. Their idea was to weigh these channels adaptively as they were not equally relevant for each tracked object. An advantage was that this enhanced layer could be used with other Siamese architectures, too. The backbone of the RASNet architecture was an Hourglass-like CNN model [97].

Li et al. [59] also remarked that different feature channels should not contribute equally to the tracking. However, the multi-level feature fusion procedure developed for their HSS-Net tracker (described in section IV-C2) made them to do so. To resolve this, their framework also utilized a simple channel attention network to adaptively assign weights to different feature channels. Its components were the global pooling layer followed by two fully connected layers. To achieve $(0, 1)$ interval for the output weighting coefficients, they added sigmoid non-linearity at the end of the network.

V. EXPERIMENTAL COMPARISON

In this chapter, we will compare the trackers in terms of their performance and design decisions in the training phase. So far, we have discussed their strengths and contributions to various problems concerning Siamese architectures. However, we do consider it important to provide results of various quantitative evaluations for overall comparison. Some surveys complement our discussion, e.g., [1], [17], [98], that contain rich quantitative comparisons of these tracking algorithms. Despite this, we provide a comparison to show the performance of the methods relevant to this survey and to make an overview of their main characteristics. Furthermore, we compare some of the important recent methods. The majority of these methods have been published in the last 4 years. This comparison is shown in two tables. Table 1 contains results of evaluation on standard visual tracking benchmarks. Table 2 summarizes contributions of each tracker and basic aspects of their training. At the end of this section, we provide a discussion about the obtained results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

A. METHODOLOGY

One of the conclusions of the [1] survey was that tracking algorithms were difficult to compare. In retrospect, we fully agree with this statement. Siamese trackers are difficult to compare with each other, even though many of them are similar in general principles. There are various benchmarks and researchers are interested in pursuing intriguing challenges, so they often evaluate their algorithms on diverse and newly published benchmarks rather than the already established ones [1]. On the other hand, even if they use the standards benchmarks, they may dive into different versions.

Table 1 shows only the results of the benchmarks we identified as the most prevailing. Even though there are more metrics available in the given benchmarks, we decided only on the listed ones because the authors used them most often. Several research papers evaluated their trackers on specific benchmarks but used different metrics. Since we wanted to make Table 1 as rich as possible, we also searched for evaluations of trackers in different papers, not just the original ones. Sometimes the search was successful and we marked the corresponding values obtained from different sources with a superscript. We do have to note that the values occasionally varied and the ordering of trackers could be different even on the same benchmark. In the case of multiple sources with different scores, we chose the most recent ones. Nevertheless, we prioritized the results published in the original paper.

For the VOT challenges (years 2015-2018 [99], [100], [101], [102]) we collected the accuracy (A), robustness (R), and expected average overlap (EAO) scores. A and R are defined as two measures for probing the tracking performance, with EAO being an overall representative of both, and is proposed as the primary measure that combines the two aspects of tracking performance [103]. As far as the Object Tracking Benchmark (OTB) (both 2013 [6] and 2015 [104] versions) is concerned, we only collected the area under curve (AUC) of success plots, even though these benchmarks provide precision plots, too. In the precision plot, a frame is marked as successful if the distance between the centers of the predicted and the ground truth BBOXes is under a given threshold [6]. However, the success plot is generally considered to be more accurate, even though both metrics measure the percentage of successfully tracked frames. Each of these plots is generated by varying the threshold values. For our purposes, we consider the sole AUC of the success plot score sufficient. During the official evaluation, tracking algorithms are ranked based on the AUC score for the success plot and precision at a threshold equal to 20 [28]. Concerning the GOT-10k [105] benchmark, we only provide the average overlap (AO) score for the same reason why we just report a success plot instead of a precision plot for OTB benchmarks.

Here is a brief explanation of how the metrics that are crucial to our discussion are computed:

- *Intersection-over-union (IoU).* A measure of a relative overlap between two image regions, i.e., if $\mathbf{b}_1$ and $\mathbf{b}_2$ are two BBOXes, then their IoU is computed as

$$\mathrm{IoU}\left(\mathbf{b}_1, \mathbf{b}_2\right) = \frac{\mathrm{area}\left(\mathbf{b}_1 \cap \mathbf{b}_2\right)}{\mathrm{area}\left(\mathbf{b}_1 \cup \mathbf{b}_2\right)}. \tag{8}$$

- *Success plot.* A frame is successfully tracked if the predicted BBOX and the ground truth BBOX have an IoU score larger than a given threshold [6].
- *Area under curve (AUC).* The area delineated by the curve of a specific plot (e.g., success or precision plot), measured on the $\langle 0, 1 \rangle$ interval in this case.
- *Accuracy (A).* The average IoU score between the predicted and ground truth BBOXes during successful tracking periods [103].
- *Robustness (R).* Measures how many times the tracker loses the target (fails) during tracking [103].
- *Expected average overlap (EAO).* This measure combines the raw values of per-frame accuracies and failures. Consider a tracking sequence of N_s frames in length. A tracker is initialized at the beginning and left to track until the end of the sequence. The performance for the given sequence (including zero-overlaps after failure) is computed as

$$\Phi_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} \Phi_i, \tag{9}$$

  where Φ_i is the per-frame overlap. To obtain the expected average overlap $\hat{\Phi}_{N_s}$, we average multiple Φ_{N_s} values over a large set of N_s-long sequences, such as

$$\hat{\Phi}_{N_s} = \langle \Phi_{N_s} \rangle. \tag{10}$$

  These computed measures are evaluated over all sequence lengths, i.e., for $N_s = 1, \ldots, N_{\max}$, which produces an *expected average overlap curve*. To estimate the final EAO score $\hat{\Phi}$, the sequence length probability density function (PDF) is needed to establish a range of typical sequence lengths for the entire benchmark. This interval, bounded by sequence lengths N_l and N_h, delineates the region for evaluation, thus

$$\hat{\Phi} = \frac{1}{N_h - N_l} \sum_{N_s=l}^{h} \hat{\Phi}_{N_s}. \tag{11}$$

  The boundaries are found as the closest points on the left and right side of the mode, such that

$$p\left(N_l\right) \approx p\left(N_h\right), \tag{12}$$

  for which the integral of the PDF is equal to 0.5 [99].
- *Average overlap (AO).* Measures an average of overlap rates between tracking results and the ground truth BBOXes over all frames [105]. It is similar to A metric defined above.

B. DISCUSSION

In this section, we elaborate on two main aspects of our analysis. First, we analyze the quantitative results on established benchmarks collected of the surveyed trackers collected from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

TABLE 1. Tracker evaluation scores on different benchmarks. We chose the most common benchmarks shared among the given trackers. Concerning the Object Tracking Benchmark (OTB) datasets (2013 and 2015 version), we collected area under curve (AUC) scores of success plots. For visual object tracking (VOT) challenges (years 2015-2018), we provide accuracy (A), robustness (R) and expected average overlap (EAO) scores (abbreviated as E due to space limitations). GOT-10k [105] (shortened as G10k) is represented by the average overlap (AO) quantities. First, second and third scores in each category are marked in **red**, **blue** and **bold**, respectively. Values marked with a superscript number were obtained from other than the original paper. Alternative sources of data were: a [55], b [41], c [16], d [98], e [63], f [40], g [57], h [86], i [56], j [69], k [20], l [106].

Tracker	OTB13	OTB15	VOT15			VOT16			VOT17			VOT18			G10k
	↑AUC		↑A	↓R	↑E	↑A	↓R	↑E	↑A	↓R	↑E	↑A	↓R	↑E	↑AO
AO-SiamRPN	0.676	**0.679**	-	-	-	0.620	0.233	0.404	-	-	-	0.581	0.286	0.364	-
CFNet	0.611[c]	-	0.560[d]	2.520[d]	-	-	-	-	-	-	-	-	-	-	0.293[k]
DaSiamRPN	-	0.658	**0.630**	**0.660**	-	0.610	0.220	**0.411**	**0.560**	**0.340**	**0.326**	0.586[h]	**0.276**[h]	0.383[h]	0.444[l]
DensSiam	-	-	**0.619**	**1.240**	**0.340**	0.560	1.080	0.331	**0.540**	**0.350**	0.250	0.456[i]	-	0.173[i]	-
DSiam	0.642	-	0.541	-	0.280	-	-	-	-	-	-	0.506[i]	-	0.195[i]	-
FCAF	-	0.649	-	-	-	0.581	1.020	0.356	-	-	-	-	-	-	-
FIGSiam	**0.679**	-	-	-	-	0.580	-	0.340	-	-	-	0.501	-	0.339	-
FPSiamRPN	-	**0.662**	-	-	-	0.609	0.670	0.354	-	-	-	0.596	0.302	0.363	-
GOTURN	0.447[a]	0.410[b]	0.512[a]	-	0.204[a]	-	-	-	-	-	-	-	-	-	0.347[l]
LSSiam	0.663	-	-	-	-	0.530	1.020	0.294	-	-	0.229	-	-	-	-
RASNet	0.670	0.642	-	-	0.327	-	-	-	-	-	**0.281**	-	-	-	-
SA-Siam	**0.677**	-	0.590	1.260	0.310	0.540	1.080	0.291	0.500	0.459	0.236	-	-	-	-
SE-SiamFC	**0.680**	0.660	-	-	-	0.590	0.240	0.360	**0.540**	**0.380**	0.270	-	-	-	-
SiamBAN	-	-	-	-	-	-	-	-	-	-	-	**0.597**	**0.178**	**0.452**	-
SiamCAR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.569**
SiamFC	0.608[a]	0.582[b]	0.550[c]	1.580[c]	0.290[c]	0.530[j]	0.870[j]	0.289[j]	0.502[c]	0.585[c]	0.188[c]	0.498[i]	-	0.188[i]	0.374[k]
SiamFF	-	0.655	-	-	-	0.512	-	0.390	-	-	-	-	-	-	-
SiamMask	-	-	-	-	-	**0.639**	**0.214**	**0.433**	-	-	-	**0.609**	**0.276**	0.380	-
SiamMask-E	-	-	-	-	-	**0.645**	**0.210**	**0.452**	-	-	-	**0.627**	**0.248**	**0.427**	-
SiamR-CNN	-	-	-	-	-	-	-	-	-	-	-	**0.609**	0.220	**0.408**	**0.649**
SiamRM	0.638	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SiamRPN	0.658[g]	0.636[e]	0.580	**1.130**[f]	**0.349**[f]	0.560	0.260[f]	0.344[f]	0.490[f]	0.460[f]	0.244[f]	0.586[j]	**0.276**[j]	0.383[j]	0.367[l]
SiamRPN++	0.650[h]	**0.696**	-	-	-	**0.642**	**0.196**	-	-	-	-	0.576[h]	0.290[h]	0.352[h]	**0.517**[k]
SiamVGG	0.665	-	**0.601**	-	**0.373**	0.564	-	0.351	**0.525**	-	**0.286**	0.527[i]	-	0.287[i]	-
SINT	0.655	0.592[b]	-	-	-	-	-	-	-	-	-	-	-	-	-
YCNN	0.601	-	-	-	-	-	-	-	-	-	-	-	-	-	-

various papers and provide our observations. Second, we discuss the primary contributions of each tracker, its backbone architecture, the presence or absence of pretraining, and training datasets.

Nevertheless, before discussing our analysis, we have to emphasize that Siamese trackers are a research direction in VOT with great potential. This branch of trackers belongs to the fastest with the accuracy-to-speed ratio being their strength. Even though we do not provide explicit values for the FPS specifically for each tracker (since they depend on hardware), we will just briefly comment on this aspect. We can generally claim that Siamese trackers operate at real-time speed, often at speeds far exceeding real-time, for instance SiamFC [12] with 94 FPS, SiamRPN [61] with 165 FPS, or LSSiam [57] with 100 FPS, to name a few. Nevertheless, there are exceptions to the rule, for example SINT [16] tracker with 4 FPS or SiamRM [29] with 21 FPS. Given this vast difference between the real-time threshold and the actual processing time, we see this as a potential to make the tracker even more accurate and robust at the cost of slightly reducing FPS while still being above the real-time threshold. But we noticed that fast trackers are also among the accurate ones (again, with existing exceptions). High processing speed is simply an inherent property of Siamese architectures.

The presence of distractors is in this paper often referred to as one of the leading causes of problems for the Siamese trackers. Considering the evaluation scores in Table 1, we conclude that this might be the case. The scores indicate that trackers where the presence of semantic background is explicitly treated often yield the top performance. Or, if the candidates for possible object locations are deliberately chosen, e.g., using RPNs. We encourage the reader to also use the Table 2 when assessing the results to remind the distinctive features of each tracker.

Considering the OTB 2013 scores, we see that the best performing tracker is SE-SiamFC. However, we also have to emphasize very similar performance of FIGSiam [56], SA-Siam [16], and AO-SiamRPN [86]. There are two reasons.

First, the runner-up in terms of the AUC score is behind by 0.001. Second, authors of SE-SiamFC [23] provided their score rounded to 2 decimal places, so the results could be slightly different. But, as will be shown later, we lack evaluations for different trackers that could potentially change the relative order. The FIGSiam and SA-Siam focused explicitly on appearance features. The AO-SiamRPN employed Kalman filtering [87] to address motion prediction and thus delineated a better region in the future frame to search for the exemplar. The FIGSiam directly aimed at the foreground and semantic background discrimination by positive pairs sampling strategy and also adopted a template updating mechanism. The SA-Siam model contained two branches, semantic and appearance branches. SE-SiamFC is practically a scale-equivariant extension to any Siamese architecture. This, again, is related to appearance features.

On the OTB 2015 benchmark, SiamRPN++ is the best performing tracker. We venture to claim that from architectural standpoint, the RPN empowers the tracker with great accuracy. A holistic glance on the highlighted cells in Table 1 shows that many top-performing trackers have RPN in their name.

When it comes to VOT challenges, the 2015 version has one outstanding result, specifically the DaSiamRPN [40]. Looking back at the definition of the R metric, the initial hypothesis may be that the robustness of this tracker is a consequence of the proposed distractor-aware module. Simply put, if the objective is not to lose track of the object, whether as a result of occlusion or the presence of similar-looking objects, the tracker needs to explicitly deal with such situations. Siamese metric learning is powerful enough to encompass numerous visual variations [107], but in case there are distractors present, then additional steps conditionally executed seem to contribute positively.

However, we should not attribute all the credit for robustness to the distractor-awareness itself. Looking at the VOT 2016 and 2018 scores we see that SiamRPN [61] and SiamMask [2], especially their derived successors, achieve even better performance. The robustness category for the VOT 2017 version is also won by the DaSiamRPN tracker, but there are missing many scores for trackers that could influence the ranking. By and large, region proposal [108] strategy is the common ground for these trackers and thus we conjecture that they have their share of the resulting leading performance. The goal of this strategy is to train the model to regress the possible locations of the exemplar in the image in advance, which may, among other things, effectively omit semantic background, thus increasing the robustness by directly addressing the drift problem. However, the RPN-based trackers come with the cost of difficult hyperparameter setup and unstable training. There is the SiamBAN tracker the authors of which explicitly avoided the use of RPN for the reasons stated above. The results of their work indicate that a tracker may perform well even without region proposals. It was also demonstrated by a very similar SiamCAR tracker. But we have to stress the lack of benchmark evaluations of these two trackers. Both works provide only one benchmark evaluation that suits our requirements. Based on the presented results, we still consider RPN-based trackers very robust and currently one of the best performing approaches. Nevertheless, these results should be taken only as a rough overview, not an accurate comparison.

One of the newest benchmarks is the GOT-10k dataset. We can see a sound performance of SiamR-CNN. We attribute the success of this tracker to a combination of the RPN-based two-stage detection with the optimal assignment of object re-detections using dynamic programming. Last but not least, the authors also utilized hard negative mining during the training to suppress similar interference. Due to the novelty of this benchmark, there are still many evaluations of previous works absent for better comparison.

The standard trend is to train the tracker on datasets that cover different domains yet try to be as general as possible to achieve good generalization ability. General object trackers should be applicable across various domains, so they are evaluated in that regard. Since Siamese trackers may be practically considered as "template matchers", then object detection datasets are also suitable for training. A common strategy is to extract these patches from various parts of the frame sequence belonging to a specific object. The aim is to extract patches from the scenes where the object is subjected to various visual disruptions, such as lightning, scale, and rotation variations, and occlusion [12]. We emphasize that the classes in the training dataset and testing dataset should never be overlapped. Before training, it is always a good practice to reach out for the benchmark documentation and exclude appropriate sequences from popular training datasets.

A prominent dataset for training Siamese models is ILSVRC 2015 [38]. This dataset contains various video sequences of general objects (a subset of the objects contained in the ImageNet dataset [49] itself). It contains approximately 4 500 videos for a total of approximately one million annotations. The included scenarios are different from other tracking benchmarks. It evaluates algorithms for object detection and image classification at a large scale. Some sequences may contain multiple objects, but this dataset is primarily used for single object tracking. Most of the time, only pairs of image patches are required in each iteration. Another new dataset is Youtube-BB [109]. It is a large-scale dataset containing video URLs with densely sampled single-object BBOXes. It consists of approximately 380 000 videos lasting for about 15-20 seconds. The videos were automatically selected to feature objects in natural settings without any editing or post-processing. Their quality is often akin to that of a hand-held cell phone camera. One more important dataset for training and evaluation of generic object tracking is GOT-10k [105]. It contains more than 10 000 video segments of real-world moving objects, covers a majority of 560+ classes of real-world moving objects, and 80+ classes of motion patterns. This dataset comes with its train, validation as well as the test set.

Table 2 shows that trackers were usually inherited from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

AlexNet [15] or ResNet [64] architectures. Authors sometimes experimented with VGGNet [71] in their ablation studies, too. It is important to mention that the discussed frameworks did not usually utilize the entire AlexNet architecture. Most of the time it was just the first few layers. For instance, GOTURN [64] used all the layers from the AlexNet up to pool-5. SiamFC [12] was inherited from AlexNet with the first five convolutional layers. Likewise, DSiam [55] adopted the branches either from AlexNet or VGG up to pool-5. Additionally, authors often tweaked the architecture to suit their needs, for example reducing the number of pooling layers for better resolution of the response maps [29]. Currently, the use of ResNet backbones (residual learning) has become popular in the deep learning community, even in other areas apart from object tracking. Table 2 shows that custom backbone architectures trained from scratch are not so prevalent.

Speaking of training, having pre-trained models saves a considerable amount of time during the training. It is a known fact that the convolutional layers that are closer to the input of the model capture local low-level features such as edges and blobs. These features can be reused across different tasks in a process called transfer learning. Only the last layers of the model capture task-specific, high-level features [13]. Visual tracking is no exception. However, as we also emphasized in section IV-A, models pre-trained on ImageNet are trained for the classification task, and tracking is more about prediction rather than pure classification [58]. Notwithstanding the objections, our analysis shows that this trend does not seem to change. Authors rather develop strategies to mitigate the potential negative consequences of adopting already pre-trained models.

VI. CONCLUSION AND FUTURE WORK

Our survey covered a specific subset of deep learning-based visual object trackers called Siamese trackers. This branch of tracking approaches is built upon principles of similarity learning. Overall, Siamese trackers aim to learn a metric embedding in a specific n-dimensional space where the distance between the embedded objects of interest reflects properties defined within the dataset during the training. We covered various aspects of Siamese trackers, and the main goal was to tackle the core traits and challenges of Siamese frameworks and to provide an overview of how existing research addresses them. We did our best to identify crucial components of Siamese models that the research either focuses on or could in the future. Nevertheless, we affirmatively highlight the comment of [1] from 2017 that objective reproducibility and comparability of results is of paramount importance, yet with current VOT methods and evaluation standards, it is difficult.

The branch of Siamese tracking brought new design principles to tracking algorithms. We primarily focused on trackers that employed the cross-correlation operation, introduced in [12]. We explored its properties and discussed that recent research does not adopt the original single-channel version of the response map. It was argued that a single channel did not encompass enough information [20], thus **multi-channel cross-correlation** layers were used [63]. Once multiple channels are present, we see an emerging trend in using **attention mechanisms** of various kinds for better feature selection [41]. The utilization of cross-correlation has a great share of the leading performance in terms of their **speed-to-accuracy ratio**.

Table 2 highlights the general trend that AlexNet-like architectures [15] are slowly subsiding, and the community has started to exploit **deeper and wider backbones** (e.g., ResNet [64]) that include padding in their convolutional layers (section IV-A). As far as backbones are concerned, various feature extraction approaches have been developed. Numerous works have demonstrated that **multi-level feature fusion** is essential for coping with similar interference. One drawback of pioneering Siamese trackers was the use of the high-level features in cross-correlation operation. However, the high-level features are satisfactory for inter-class discrimination, but not for handling minor yet relevant differences between intra-class distractors. The notion of coalescing features from multiple levels using diverse approaches has become a standard (section IV-C2).

Because of the obtained results and knowledge from the reviewed literature, we see **distractors** (**similar interference**) and severe occlusion as the primary cause of problems with Siamese architectures. The majority of the discussed trackers attempted to tackle these issues in various ways, from the adoption of region proposals [108] through explicit **distractor-awareness** [40] to conditional **re-detection** [29], [77]. After all, these problems are still not completely solved and pose a challenge to Siamese trackers as they are prone to drift to the semantic background.

Numerous works have commented that including memory or **template updating** strategies could improve the performance [12], [57]. It seems that relying purely on the initial frame may cause the tracker to fail when the object undergoes severe visual deformations. If template updating (which may be time-consuming) is not employed, then, we think, other mechanisms such as robust region proposals seem to have a significantly positive impact on performance, as demonstrated by trackers [61], [63], [40], [69].

In Table 1, we provided a recent quantitative comparison of surveyed trackers on established benchmarks. We can see that the top-performing trackers are based on **region proposals** [61], [63], [69], [86]. The comprehensive survey from Marvasti-Zadeh et al. [17] also reached a similar conclusion. We can also see in Table 2 that approaches that explicitly deal with semantic background or adopt multi-level feature fusion are among the top-performing trackers, too. One-shot learning is also similar in principle [73]. The idea of having the object BBOX provided only once during the initialization phase and then having to identify it again as it undergoes various transformations is tantamount to one-shot learning. To support this, the SiamRPN tracker was directly formulated as one-shot learning [61]. We believe that this line of research may provide useful insights into tracking itself. Nonetheless,

learning the visual object model from a single appearance is an ill-posed problem and requires adequate data preparation to achieve reasonable generalization ability [13].

Despite the appreciation of RPN above, they come with several obstacles that forced the community to circumvent them. One disadvantage is that it is very sensitive to hyper-parameter setup, e.g., object scale or aspect ratio. Several modern trackers deliberately avoided anchor boxes that stand at the core of region proposals. Specifically, tracking was framed as **pixel-wise classification and regression**, making it an **anchor-free** solution that has become popular [20], [88].

Siamese networks often come in a form of a "Y-shaped" network. Similarity learning started as signature verification [18], and nowadays it is also largely used in the area of object **re-identification** (ReID). Face ReID is a well-researched topic [45], [42], but also vehicle ReID has recently become a target for researchers [47], [110]. Tracking and object ReID face similar challenges and benefit from learning metric spaces that handle various appearance distortions of the object of interest [111]. In this area, researchers have started to employ not only contrastive (pairwise) loss but triplet loss [42], too. Even quadruple loss [112]. Some works focused on Siamese trackers and attempted to use the triplet loss, such as [113]. Moreover, pair or triplet mining strategies to support training the model on incrementally harder samples as the training advances were also adopted in ReID [42]. We did not frequently encounter this method in tracking literature, but there were few exceptions, e.g., [81].

The use of **recurrent neural network** (RNN) may help by incorporating more contextual information about the object [84] or to model its appearance better which leads to improved performance in situations with similar interference [58]. This module may be used to incorporate **temporal information** [114], too. We believe that Siamese trackers, due to the way they are trained, are stripped away of temporal information, and the computation of the similarity score is then performed independently of how many frames apart the two embeddings were produced. Some trackers attempt to model the appearance of the targets in the long term by applying RNN, but the decay of the object's features exacerbates the tracking performance [115]. Even though RNNs may be useful for capturing temporal information between video frames, they are limited in their stability as well as learning long-term dependencies [17]. But still, Zhao et al. [115] proposed a special anti-decay long short-term memory module for Siamese trackers. Nevertheless, the use of RNNs seems to be more dominant in multi-object tracking [116].

Reinforcement learning (RL) is a large area rich with potential for broad applications. In the context of Siamese tracking, the work of [117] introduced RL to tackle online model updating because a fixed template was not effective enough to capture target appearance variations. They collected a series of templates and then used the actor-critic framework to learn how to maintain them. Another dynamic aspect of visual tracking is trajectory prediction. Abdelpakey et al. [118] utilized a RL approach based on dynamic policy gradient to produce a continuous action that predicted the optimal object location.

Even though our work focused on single object tracking, there are emerging papers where Siamese architectures such as SiamFC [12] were integrated into a **multi-object tracking** pipeline. Shuai et al. [119] proposed one of such frameworks that are capable of handling tracking, detection, and ReID at the same time while allowing the utilization of any Siamese tracker. Another simple and very effective extension of SiamFC tracker was to use n exemplars to produce n response maps and thus perform tracking of n objects simultaneously [120]. We believe this paradigm of tracking is yet to uncover its full potential.

Siamese networks have yet to be fully understood [107], [1], [35]. It is important to consider their inherent properties and exploit them, as many of the discussed papers have demonstrated. In this work, we attempted to underline the current state-of-the-art solutions of the Siamese-based visual object trackers and the challenges they face.

REFERENCES

[1] R. Pflugfelder, "An in-depth analysis of visual tracking with siamese neural networks," arXiv preprint arXiv:1707.00569, 2017.
[2] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1328–1338, 2019.
[3] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 7, pp. 1442–1468, 2014.
[4] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," ACM Comput. Surv., vol. 38, no. 4, p. 13–es, 2006.
[5] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," ACM Trans. Intell. Syst. Technol., vol. 4, no. 4, 2013.
[6] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2411–2418, 2013.
[7] A. Emami, F. Dadgostar, A. Bigdeli, and B. C. Lovell, "Role of spatiotemporal oriented energy features for robust visual tracking in video surveillance," in 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance, pp. 349–354, 2012.
[8] L. Liu, J. Xing, and H. Ai, "Multi-view vehicle detection and tracking in crossroads," in The First Asian Conference on Pattern Recognition, pp. 608–612, 2011.
[9] L. Liu, J. Xing, H. Ai, and X. Ruan, "Hand posture recognition using finger geometric feature," in Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pp. 565–568, IEEE, 2012.
[10] D. Held, D. Guillory, B. Rebsamen, S. Thrun, and S. Savarese, "A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues.," in Robotics: Science and Systems (D. Hsu, N. M. Amato, S. Berman, and S. A. Jacobs, eds.), 2016.
[11] T. Sikora, "The mpeg-4 video standard verification model," IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 19–31, 1997.
[12] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in Computer Vision – ECCV 2016 Workshops (G. Hua and H. Jégou, eds.), (Cham), pp. 850–865, Springer International Publishing, 2016.
[13] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in 2015 IEEE International Conference on Computer Vision (ICCV), pp. 3119–3127, 2015.
[14] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep learning, vol. 1. MIT press Cambridge, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.
[16] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4834–4843, 2018.
[17] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei, "Deep learning for visual tracking: A comprehensive survey," IEEE Transactions on Intelligent Transportation Systems, pp. 1–26, 2021.
[18] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," International Journal of Pattern Recognition and Artificial Intelligence, vol. 7, no. 04, pp. 669–688, 1993.
[19] K. Chen and W. Tao, "Once for all: A two-flow convolutional neural network for visual tracking," IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 12, pp. 3377–3386, 2018.
[20] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, "Siamcar: Siamese fully convolutional classification and regression for visual tracking," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6269–6277, 2020.
[21] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in Proceedings of the IEEE international conference on computer vision, pp. 9627–9636, 2019.
[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440, 2015.
[23] I. Sosnovik, A. Moskalev, and A. W. Smeulders, "Scale equivariance improves siamese tracking," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2765–2774, 2021.
[24] A. S. Jalal and V. Singh, "The state-of-the-art in visual object tracking," Informatica, vol. 36, no. 3, 2012.
[25] S. Dubuisson and C. Gonzales, "A survey of datasets for visual tracking," Machine Vision and Applications, vol. 27, pp. 23–52, Jan 2016.
[26] V. Ramalakshmi and M. Alex, "A survey on visual object tracking: Datasets methods and metrics," Int. Res. J. Eng. Technol., vol. 3, no. 11, pp. 1–6, 2016.
[27] S. You, H. Zhu, M. Li, and Y. Li, "A review of visual trackers and analysis of its application to mobile robot," arXiv preprint arXiv:1910.09761, 2019.
[28] R. Tao, E. Gavves, and A. M. Smeulders, "Siamese instance search for tracking," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (Los Alamitos, CA, USA), pp. 1420–1429, IEEE Computer Society, jun 2016.
[29] D. Li, Y. Yu, and X. Chen, "Object tracking framework with siamese network and re-detection mechanism," EURASIP Journal on Wireless Communications and Networking, vol. 2019, p. 261, Nov 2019.
[30] S. Salti, A. Cavallaro, and L. Di Stefano, "Adaptive appearance modeling for video tracking: Survey and evaluation," IEEE Transactions on Image Processing, vol. 21, no. 10, pp. 4334–4348, 2012.
[31] C. Li, B. Yang, and C. Li, "Deep learning based visual tracking: A review," DEStech Transactions on Computer Science and Engineering, 07 2017.
[32] M. Y. Abbass, K.-C. Kwon, N. Kim, S. A. Abdelwahab, F. E. A. El-Samie, and A. A. Khalaf, "A survey on online learning for visual tracking," VISUAL COMPUTER, 2020.
[33] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung, "Handcrafted and deep trackers: Recent visual object tracking approaches and trends," ACM Comput. Surv., vol. 52, no. 2, 2019.
[34] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler, "Learning by tracking: Siamese cnn for robust target association," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 33–40, 2016.
[35] S. Roy, M. Harandi, R. Nock, and R. Hartley, "Siamese networks: The tale of two manifolds," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3046–3055, 2019.
[36] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, 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, no. 11, pp. 2137–2155, 2016.
[37] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," arXiv preprint arXiv:1501.04587, 2015.
[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., "Imagenet large scale visual recognition challenge," International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.
[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, vol. 323, no. 6088, pp. 533–536, 1986.
[40] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in Computer Vision – ECCV 2018 (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 103–119, Springer International Publishing, 2018.
[41] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: Residual attentional siamese network for high performance online visual tracking," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4854–4863, 2018.
[42] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," arXiv preprint arXiv:1703.07737, 2017.
[43] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 2, pp. 1735–1742, 2006.
[44] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in ICML deep learning workshop, vol. 2, Lille, 2015.
[45] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815–823, 2015.
[46] H. Wu, Z. Xu, J. Zhang, W. Yan, and X. Ma, "Face recognition based on convolution siamese networks," in 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 1–5, 2017.
[47] R. Kuma, E. Weill, F. Aghdasi, and P. Sriram, "Vehicle re-identification: an efficient baseline using triplet embedding," in 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–9, 2019.
[48] S. Zagoruyko and N. Komodakis, "Deep compare: A study on using convolutional neural networks to compare image patches," Computer Vision and Image Understanding, vol. 164, pp. 38–55, 2017.
[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255, Ieee, 2009.
[50] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in European Conference on Computer Vision, pp. 749–765, Springer, 2016.
[51] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv preprint arXiv:1312.4400, 2013.
[52] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9, 2015.
[53] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 539–546 vol. 1, 2005.
[54] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2805–2813, 2017.
[55] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in 2017 IEEE International Conference on Computer Vision (ICCV), pp. 1781–1789, 2017.
[56] D. Li and Y. Yu, "Foreground information guidance for siamese visual tracking," IEEE Access, vol. 8, pp. 55905–55914, 2020.
[57] Z. Liang and J. Shen, "Local semantic siamese networks for fast tracking," IEEE Transactions on Image Processing, vol. 29, pp. 3351–3364, 2020.
[58] H. Fan and H. Ling, "Sanet: Structure-aware network for visual tracking," in 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2217–2224, 2017.
[59] X. Li, Q. Liu, N. Fan, Z. He, and H. Wang, "Hierarchical spatial-aware siamese network for thermal infrared object tracking," Knowledge-Based Systems, vol. 166, pp. 71–81, 2019.
[60] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4917–4926, 2019.

[61] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8971–8980, 2018.

[62] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, pp. 91–99, 2015.

[63] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4282–4291, 2019.

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.

[65] M. H. Abdelpakey, M. S. Shehata, and M. M. Mohamed, "Denssiam: End-to-end densely-siamese network with self-attention model for object tracking," in Advances in Visual Computing (G. Bebis, R. Boyle, B. Parvin, D. Koracin, M. Turek, S. Ramalingam, K. Xu, S. Lin, B. Alsallakh, J. Yang, E. Cuervo, and J. Ventura, eds.), (Cham), pp. 463–473, Springer International Publishing, 2018.

[66] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[67] G. Han, H. Du, J. Liu, N. Sun, and X. Li, "Fully conventional anchor-free siamese networks for object tracking," IEEE Access, vol. 7, pp. 123934–123943, 2019.

[68] X. Wu, X. Wang, X. Zhou, and S. Jian, "Sta: Adversarial attacks on siamese trackers," arXiv preprint arXiv:1909.03413, 2019.

[69] Y. Rao, Y. Cheng, J. Xue, J. Pu, Q. Wang, R. Jin, and Q. Wang, "Fpsiamrpn: Feature pyramid siamese network with region proposal network for target tracking," IEEE Access, vol. 8, pp. 176158–176169, 2020.

[70] Y. Li and X. Zhang, "Siamvgg: Visual tracking using deeper siamese networks," arXiv preprint arXiv:1902.02804, 2019.

[71] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[72] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in CVPR 2011, pp. 1521–1528, 2011.

[73] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in Advances in neural information processing systems, pp. 523–531, 2016.

[74] B. X. Chen and J. Tsotsos, "Fast visual object tracking using ellipse fitting for rotated bounding boxes," in 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 2281–2289, 2019.

[75] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. D. Hu, "Discriminant correlation filters network for visual tracking. arxiv 2017," arXiv preprint arXiv:1704.04057, 2017.

[76] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6469–6477, 2017.

[77] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe, "Siam r-cnn: Visual tracking by re-detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6578–6588, 2020.

[78] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708, 2017.

[79] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, pp. 5998–6008, 2017.

[80] Y. Luo, Y. Cai, B. Wang, J. Wang, and Y. Wang, "Siamff: Visual tracking with a siamese network combining information fusion with rectangular window filtering," IEEE Access, vol. 8, pp. 119899–119910, 2020.

[81] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2999–3007, 2017.

[82] Q. Liu, X. Li, Z. He, N. Fan, D. Yuan, and H. Wang, "Learning deep multi-level similarity for thermal infrared object tracking," IEEE Transactions on Multimedia, pp. 1–1, 2020.

[83] Q. Liu, X. Li, Z. He, N. Fan, D. Yuan, W. Liu, and Y. Liang, "Multi-task driven feature models for thermal infrared tracking," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 11604–11611, Apr. 2020.

[84] X. Xu, B. Ma, H. Chang, and X. Chen, "Siamese recurrent architecture for visual tracking," in 2017 IEEE International Conference on Image Processing (ICIP), pp. 1152–1156, 2017.

[85] D. K. Gupta, E. Gavves, and A. W. Smeulders, "Tackling occlusion in siamese tracking with structured dropouts," arXiv preprint arXiv:2006.16571, 2020.

[86] F. Wu, J. Zhang, and Z. Xu, "Stably adaptive anti-occlusion siamese region proposal network for real-time object tracking," IEEE Access, vol. 8, pp. 161349–161360, 2020.

[87] T. Basar, A New Approach to Linear Filtering and Prediction Problems, pp. 167–179. Wiley-IEEE Press, 2001.

[88] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese box adaptive network for visual tracking," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6667–6676, 2020.

[89] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 834–848, 2018.

[90] I. Sosnovik, M. Szmaja, and A. Smeulders, "Scale-equivariant steerable networks," arXiv preprint arXiv:1910.11093, 2019.

[91] D. K. Gupta, D. Arya, and E. Gavves, "Rotation equivariant siamese networks for tracking," arXiv preprint arXiv:2012.13078, 2020.

[92] Y. Zha, M. Wu, Z. Qiu, S. Dong, F. Yang, and P. Zhang, "Distractor-aware visual tracking by online siamese network," IEEE Access, vol. 7, pp. 89777–89788, 2019.

[93] L. Zhang, A. Gonzalez-Garcia, J. v. d. Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for siamese trackers," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4010–4019, 2019.

[94] Z. Xu, H. Luo, B. Hui, Z. Chang, and M. Ju, "Siamese tracking with adaptive template-updating strategy," Applied Sciences, vol. 9, no. 18, 2019.

[95] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[96] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in European conference on computer vision, pp. 391–405, Springer, 2014.

[97] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in European conference on computer vision, pp. 483–499, Springer, 2016.

[98] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," Pattern Recognition, vol. 76, pp. 323–338, 2018.

[99] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2015 challenge results," in Proceedings of the IEEE international conference on computer vision workshops, pp. 1–23, 2015.

[100] S. Hadfield, R. Bowden, and K. Lebeda, "The visual object tracking vot2016 challenge results," Lecture Notes in Computer Science, vol. 9914, pp. 777–823, 2016.

[101] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey, et al., "The visual object tracking vot2017 challenge results," in Proceedings of the IEEE international conference on computer vision workshops, pp. 1949–1972, 2017.

[102] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. ˇCehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, et al., "The sixth visual object tracking vot2018 challenge results," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 0–0, 2018.

[103] 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 Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pp. 0–0, 2019.

[104] Y. Wu, J. Lim, and M. Yang, "Object tracking benchmark," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1834–1848, 2015.

[105] L. Huang, X. Zhao, and K. Huang, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," IEEE Transactions on Pattern Analysis and Machine Intelligence, p. 1–1, 2019.

[106] "Got10k leaderboard." http://got-10k.aitestunion.com/leaderboard, 2021. [Online; accessed 23-June-2021].

[107] M. Zheng, S. Karanam, T. Chen, R. J. Radke, and Z. Wu, "Towards visually explaining similarity models," arXiv preprint arXiv:2008.06035, 2020.
[108] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587, 2014.
[109] 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 proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5296–5305, 2017.
[110] H. Wang, S. Sun, L. Zhou, L. Guo, X. Min, and C. Li, "Local feature-aware siamese matching model for vehicle re-identification," Applied Sciences, vol. 10, no. 7, 2020.
[111] Z. Tang, M. Naphade, M. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8789–8798, 2019.
[112] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 403–412, 2017.
[113] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 459–474, 2018.
[114] M. Kim, S. Alletto, and L. Rigazio, "Similarity mapping with enhanced siamese network for multi-object tracking," arXiv preprint arXiv:1609.09156, 2016.
[115] F. Zhao, T. Zhang, Y. Wu, M. Tang, and J. Wang, "Antidecay lstm for siamese tracking with adversarial learning," IEEE Transactions on Neural Networks and Learning Systems, 2020.
[116] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie, "Trajectory factory: Tracklet cleaving and re-connection by deep siamese bi-gru for multiple object tracking," in 2018 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6, IEEE, 2018.
[117] F. Zhao, T. Zhang, Y. Song, M. Tang, X. Wang, and J. Wang, "Siamese regression tracking with reinforced template updating," IEEE Transactions on Image Processing, vol. 30, pp. 628–640, 2021.
[118] M. H. Abdelpakey and M. S. Shehata, "Dp-siam: Dynamic policy siamese network for robust object tracking," IEEE Transactions on Image Processing, vol. 29, pp. 1479–1492, 2020.
[119] B. Shuai, A. G. Berneshawi, D. Modolo, and J. Tighe, "Multi-object tracking with siamese track-rcnn," arXiv preprint arXiv:2004.07786, 2020.
[120] L. Vaquero, M. Mucientes, and V. M. Brea, "Siammt: Real-time arbitrary multi-object tracking,"

MILAN ONDRAŠOVIČ received his B.Sc. and M.Sc. degrees from the University of Žilina in applied informatics in 2017 and 2019, respectively. Both of his final theses heavily involved computer vision. Currently, he is pursuing a Ph.D. degree at the University of Žilina. His research topic is deep machine learning applied to visual object tracking.

PETER TARÁBEK received the Ph.D. degree from the University of Žilina, in 2009. He is currently an Assistant Professor with the Department of Mathematical Methods and Operations Research, University of Žilina. His research includes artificial intelligence, deep machine learning, and computer vision. He has participated in several research and applied projects, in which he developed solutions in areas of computer vision, intelligent transportation systems, and Industry 4.0.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3101988, IEEE Access

TABLE 2. General information about surveyed trackers that we considered important. This table aims to highlight the contributions of each tracker. However, some works experimented with various settings. In such cases, we separated the possible settings within the same cell. The supplementary information about backbone and training serves for more granular comparison and emphasizes existing design trends. Trackers marked with symbol "*" deal with a specific task of TIR tracking and not standard VOT.

Tracker	Year	Characteristic	Backbone	Pre-training	Training data
AO-SiamRPN	2020	anti-occlusion Kalman filtering	ResNet	ImageNet	COCO, ILSVRC15, Youtube-BB
CFNet	2017	correlation filters	AlexNet	ImageNet	ILSVRC15
DaSiamRPN	2018	distractor awareness, sample mining strategies	AlexNet	ImageNet	ILSVRC15, Youtube-BB, COCO
DensSiam	2018	densely-connected layers, attention	custom	no	ILSVRC15
DSiam	2017	transformations: appearance variation, background suppression	AlexNet, VGG	no	ILSVRC15
FCAF	2019	anchor-free region proposal network, centerness	ResNet	ImageNet	ILSVRC15
FIGSiam	2020	positive pair sampling strategy, background padding, template updating	ResNet	no	ILSVRC15, COCO
FPSiamRPN	2020	feature pyramids, region proposal network	ResNet	ImageNet	ILSVRC15, COCO
GOTURN	2016	BBOX regression, Laplace distribution	custom	ImageNet	ImageNet14, ALOV
HSSNet *	2019	multi-level features, STN, spatial-aware network, correlation filter	AlexNet	no	ILSVRC15
LSSiam	2019	local semantic features, classification, template updating	AlexNet	ImageNet	ILSVRC15
MLSSNet *	2020	multi-level similarity, structural and semantic correlation similarity networks, adaptive ensemble module	AlexNet	no	TIR-specific
MMNet *	2019	classification, fine-grained and discriminative matching branches, CFNet utilization	AlexNet	no	TIR-specific
RASNet	2018	attention (general, residual, channel), weighted cross-correlation	custom	no	ILSVRC15
RE-SiamNet	2020	rotation-equivariance	ResNet, AlexNet	ImageNet, no	ILSVRC15, GOT-10k, Youtube-BB
SA-Siam	2016	semantic and appearance branch	AlexNet	no	ILSVRC15
SE-SiamFC	2021	scale-equivariance	AlexNet	ImageNet	ILSVRC15, GOT-10k
SiamBAN	2020	box adaptive head, classification and regression	ResNet	ImageNet	COCO, ILSVRC15, Youtube-BB, GOT-10k, LaSOT
SiamCAR	2020	pixel-wise classification and BBOX regression, centerness	ResNet	ImageNet	COCO, ILSVRC15, Youtube-BB
SiamFC	2016	fully convolutional, cross-correlation, logistic loss	AlexNet	no	ILSVRC15
SiamFF	2020	shallow-deep feature fusion, rectangular filtering	AlexNet	no	ILSVRC15, GOT-10k
SiamMask	2019	BBOX regression, binary segmentation, rotated BBOX	ResNet	ImageNet	COCO, ILSVRC15, Youtube-VOS
SiamR-CNN	2020	Re-detection with Faster R-CNN, dynamic programming for tracking history resolution	ResNet	COCO	ILSVRC15, Youtube-VOS, LaSOT, GOT-10k
SiamRM	2019	distractor detection, re-detection	AlexNet	ImageNet	ILSVRC15
SiamRPN	2018	region proposals, tracking as one-shot detection	AlexNet	ImageNet	ILSVRC15, Youtube-BB
SiamRPN++	2019	region proposals, depth-wise cross-correlation	ResNet	ImageNet	COCO, ILSVRC15, Youtube-BB
SiamVGG	2019	modified VGG backbone	VGG	no	ILSVRC15, Youtube-BB
SINT	2016	margin contrastive loss	AlexNet, VGG	ImageNet	ImageNet12, ALOV
YCNN	2017	probability map, BBOX regression	custom	no	ImageNet12, ALOV, VOT15