SiamMT: Real-Time Arbitrary Multi-Object Tracking

Lorenzo Vaquero, Manuel Mucientes and Víctor M. Brea
Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)
Universidade de Santiago de Compostela, Santiago de Compostela, Spain
Email: lorenzo.vaquero.otal@usc.es, manuel.mucientes@usc.es, victor.brea@usc.es

***Abstract*—Visual object tracking is of great interest in many applications, as it preserves the identity of an object throughout a video. However, while real applications demand systems capable of real-time-tracking multiple objects, multi-object tracking solutions usually follow the tracking-by-detection paradigm, thus they depend on running a costly detector in each frame, and they do not allow the tracking of arbitrary objects, i.e., they require training for specific classes. In response to this need, this work presents the architecture of SiamMT, a system capable of efficiently applying individual visual tracking techniques to multiple objects in real-time. This makes it the first deep-learning-based arbitrary multi-object tracker. To achieve this, we propose global frame features extraction by using a fully-convolutional neural network, followed by the cropping and resizing of the different object search areas. The final similarity operation between these search areas and the target exemplars is carried out with an optimized pairwise cross-correlation. These novelties allow the system to track multiple targets in a scalable manner, achieving 25 fps with 60 simultaneous objects for VGA videos and 40 objects for HD720 videos, all with a tracking quality similar to SiamFC.**

I. Introduction

Visual object tracking is a major component in many computer vision systems such as video surveillance [1] or autonomous vehicles [2] since it maintains the identity of an object throughout the frames of a video. Despite the recent advances in the field, there are still difficulties in dealing with factors such as deformations or fast motions. Arbitrary object trackers are class-agnostic, namely, they are capable of following objects regardless of their category, without requiring retraining or knowing the class to which they belong. This is very useful in situations where the objects of interest are defined through techniques like background subtraction [3] or visual attention models [4]. Moreover, the tracking of multiple simultaneous objects arises new challenges, such as total occlusions or identity switches. That is why, nowadays, the adopted approaches to address multiple-object and single-object tracking differ.

The most straightforward approach to tackle multi-object tracking is to instantiate many individual trackers in parallel. However, this is very inefficient and is only feasible when there are few targets on the scene. For that reason, for a few years now, multi-object tracking is done through the association of detections [5], [6]. However, since tracking by association of detections tends to be slow and requires the detector to be pretrained with the categories to be tracked —arbitrary targets are not allowed—, other techniques are often used when tracking individual objects.

Individual visual trackers work by establishing the appearance model of an arbitrarily defined object, for its comparison with each one of the frames in the video. This model can be generated from the video itself, using its own frames as examples, or in a general manner, using large databases of labeled videos. Two alternatives have dominated the state of the art in individual tracking: those based on Discriminative Correlation Filters (DCF) and those employing deep learning techniques. Generally, individual visual trackers stand out for their accuracy and speed. However, their architectures do not allow to track multiple targets, so an independent tracker must be instantiated for each followed object, causing an exponential decrease in the speed of the system.

To enable the online visual tracking of multiple arbitrary objects at real-time, this paper presents SiamMT, a novel method that extends the SiamFC [7] framework. The main novelty of our technique lies in that we set aside the current formulations in multiple object tracking through association of detections in favor of a pure tracking paradigm. Thus, differently from current state-of-the-art multiple object tracking techniques, there is no need to run a detector in each frame of the sequence, and we can apply the latest advances in single object tracking to multiple object scenarios. To the best of our knowledge, SiamMT is the first deep-learning-based arbitrary multi-object tracker. The main contributions of this work are summarized as follows:

- We propose SiamMT, a Siamese Convolutional Neural Network capable of real-time-tracking multiple arbitrary objects in a scalable manner. Its design would allow its application in SiamFC-based architectures, either by training it end-to-end or by maintaining their learned weights via a network-based transfer learning procedure [8].
- In order to allow the tracking of different sized objects in the same frame, we establish a reformulation of the RoIAlign operation [9], making it capable of cropping and resizing features extracted with a fully-convolutional backbone without padding.
- We define a new similarity operation which, based on the properties of depthwise cross-correlation, enables the efficient pairwise comparison of multiple feature maps.
- Our approach is able to track 60 simultaneous objects in

VGA video and 40 objects in HD720 video at 25 fps, all with a robustness and accuracy comparable to other single-object trackers.

II. Related Work

A. Multiple object tracking

As previously stated, the most adopted approach to multiple object tracking is the association of detections [5]. These methods, which generally lack speed due to the detectors' runtimes and do not allow the tracking of arbitrary objects, are divided into offline and online tracking systems.

Offline tracking by detection. Offline trackers by detection stand out for their accuracy and tolerance towards detector errors, with processing speed being their greatest disadvantage. Within this category, the most popular methods today are those based on graph probabilistic algorithms [10], joint probabilistic data association models [11], and appearance discrimination [12]. Here we can also find the so-called multi-class trackers [13], which can consider several types of objects at once. However, they still require additional training for each category or need to know the object's class, so they cannot be considered truly arbitrary.

Online tracking by detection. On the other hand, online trackers by detection can process streaming video using techniques of a more local nature. This allows them to achieve near-real-time processing speeds, but at the cost of decreasing their accuracy. Within this category, it is possible to find systems that make use of graph partitioning [14] or multiple hypotheses [15]. Additionally, recent deep learning-based techniques are demonstrating potent detection association capabilities with very interesting approaches, either by applying deep appearance matching models [16], by regressing the detections on the previous frame [17], or by embedding the association model in the target detector itself [18]. These types of trackers show promising for many applications, but still rely heavily on having accurate detections in all frames, thus they do not support arbitrary objects.

B. Single object tracking

Single object visual trackers define the appearance model of an object, whose location and dimensions are provided for the first frame of the sequence. This model is then compared, frame by frame, with the search region in which the target is expected to be found, in order to update its coordinates and size. This allows this kind of trackers to be very precise and fast and, as they usually implement a generic similarity function or learn it online, most of them are class-agnostic, without requiring continuous detections or retraining.

Discriminative Correlation Filters for single object tracking. DCF-based trackers predict the position of the object by training a filter that distinguishes between the element of interest and the background of the scene. Early versions focused on a single feature and represented an arbitrary target with a single filter [19]. Following this, improvements in accuracy were made with the incorporation of multidimensional features [20] and other extensions such as non-linear kernels [21]. However, these techniques have been somewhat displaced by more accurate approaches that train a similarity function offline through deep learning models.

Deep learning for single object tracking. On the other hand, deep learning-based trackers employ deep convolutional neural networks (CNNs) to train a similarity function which, starting from the initial appearance of the object, indicates its position in each frame's search area. One of the first architectures based on this concept, and the forerunner of the current state of the art, is SiamFC [7]. It uses a convolutional Siamese network for the extraction of images features, followed by a cross-correlation operator to locate the object's position. The main appeal of this concept is that it is a straightforward and accurate solution that allows high-speed tracking of an arbitrary object. This is why several contributions have been made to the original architecture, enabling the online update of the model [22], allowing the regression of the object's bounding-box [23], or even incorporating segmentation information [24]. However, most of these new approaches tend to add considerable complexity or are made at the expense of decreasing the tracking speed, so they are less suitable for extension towards multiple object tracking systems.

In summary, single object visual tracking is being dominated by deep learning and becoming more robust and accurate, yet more complex and slow. This is why we have developed our approach around SiamFC, because it is fast, simple, effective, and is the basis of most of the VOT-2019 [25] top trackers. Therefore, present and future SiamFC-based architectures can benefit from our proposal.

III. SiamMT Network Architecture

The design of SiamMT's network architecture is based on the SiamFC [7] tracker, which uses deep-learning similarity metrics to track individual objects at a high number of frames per second. The network architecture has been modified to allow the efficient tracking of multiple simultaneous objects.

A. SiamFC's network architecture

The network architecture of SiamFC is described in Fig. 1a. It obtains the target's current position by comparing its exemplar image with the current frame. Therefore, the first action required for tracking is the creation of such an exemplar image through a crop and resize operator κ. This image is extracted from the first frame where the object can be recognized, and consists of the bounding box containing the target plus a context margin, all scaled to an area of 127×127 pixels.

Once the exemplar image has been obtained, the frame-by-frame tracking begins, which is performed over a search area centered on the target's last known position. The extraction of this area is carried out in a very similar way to obtaining the first frame's exemplar image, but covering a larger area and scaling it to 255×255 pixels. Thus, this rescaled image always has the same proportion with respect to the size of the object, which is a key factor in this kind of architectures.

Therefore, starting from the exemplar and search area images, their features are extracted employing φ, a fully-convolutional —without padding— siamese network based

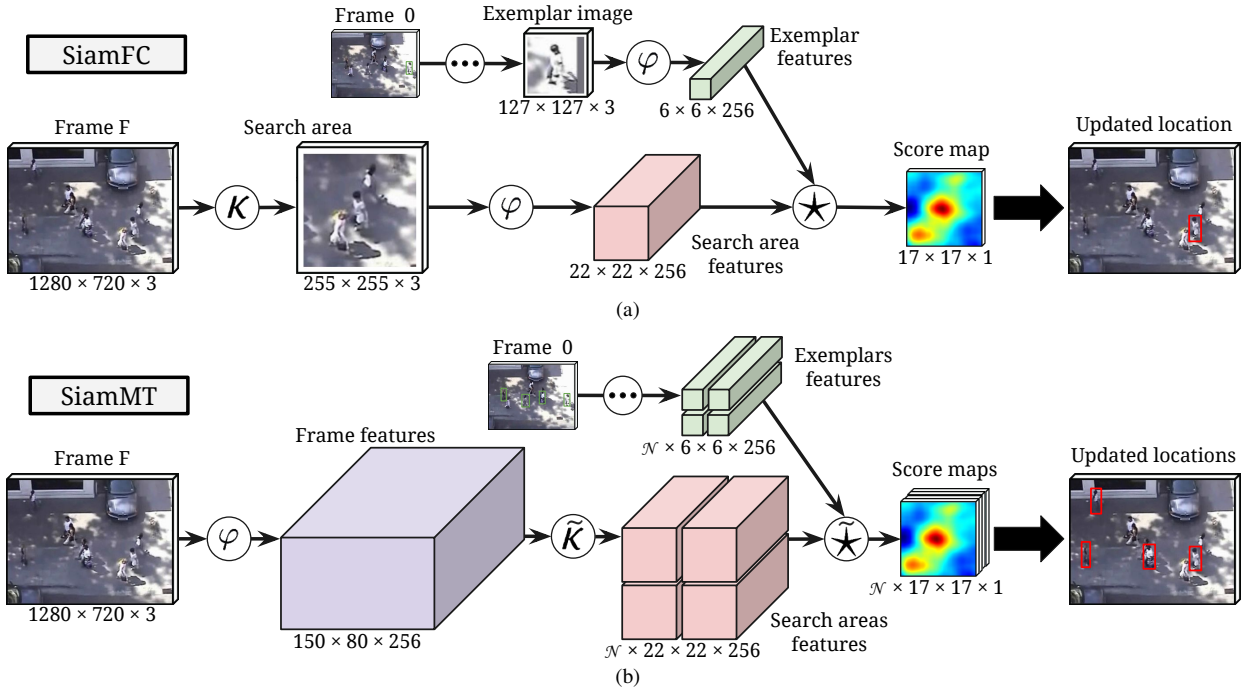

Fig. 1. SiamFC (a) and SiamMT (b) network architectures during the inference phase. SiamMT first extracts the features of the entire frame via a φ backbone. After this, the features of the various search areas are cropped and resized using the $\tilde{\kappa}$ operator, which is based on RoIAlign [9]. Finally, these features are combined with those of their respective exemplars through an optimized pairwise cross-correlation $\tilde{\star}$, obtaining score maps that indicate the new positions of the $\mathcal{N}$ objects.

on AlexNet [26]. The 256 channels obtained as a result are compared using a cross-correlation operator $\star$, generating a score map of size 17×17 that states the probability of the object being in each region of the search area. The fact that padding is not included in any of the convolutions is particularly relevant since, otherwise, the strict translation invariance property of these operations would be lost [27].

To improve the interpretation of the score map, it is upsampled to 272×272 pixels via bicubic interpolation, followed by the penalization of the regions furthest from the center. In this way, by selecting the element with the highest value from the matrix and transferring its coordinates to the input frame, it is possible to obtain the new position of the object. To allow the detection of changes in the object size, this process is carried out using a search area at 3 different scales.

B. Modifying the SiamFC architecture to multiple objects

SiamFC's network architecture was modified to support efficient multiple object tracking, as shown in Fig. 1b, giving rise to SiamMT. Its most relevant features are described below.

Global features extraction. The first modification consists of the removal of the image crop and resize module κ. This change, although it results in a greater overall computational cost for a single object —since the frames are usually larger than 255 px^2— it allows the reuse of features when there are multiple objects to follow. Such reuse is crucial for the system scalability since the computation of the features is the most expensive operation in the whole architecture and its execution per target would be unfeasible for many objects on the scene.

Cropping and resizing of features. After the global features extraction, it becomes necessary to arrange the different tensors —exemplars and frame features— for the comparison through the similarity operator. At first glance, a direct correlation of the whole frame features with those of each exemplar might seem sufficient. However, this is not feasible in an environment where the objects to be followed undergo changes of scale. The reason for this is that the main trackers through similarity [7], [22]–[24] have a relatively low tolerance for discrepancies between the object size in the exemplar image and in the search area —supporting a maximum difference of up to a 15%—. This is why the region represented in the search area must be gradually adjusted with each step of inference, increasing or decreasing it so that the size of the object is always constant. Although this is an issue that often goes unnoticed in the formulations of the different trackers, it enables them to maintain progressive changes of scale throughout a video.

To allow the correct management of the objects' scale, the approach taken in SiamMT consists of the cropping and resizing of the frame features that represent the search area of each target. These operations, while straightforward in images, are particularly challenging when carried out on features. The first proposal for carrying out this task comes with RoIPool [28], which creates an area of discrete coordinates delimiting the region of interest and divides it in sections (bins). Finally, each bin is assigned the value corresponding to the highest value of the pixels it contains, obtaining a fixed-size feature map —e.g. 6×6— from the nonuniform sized input.

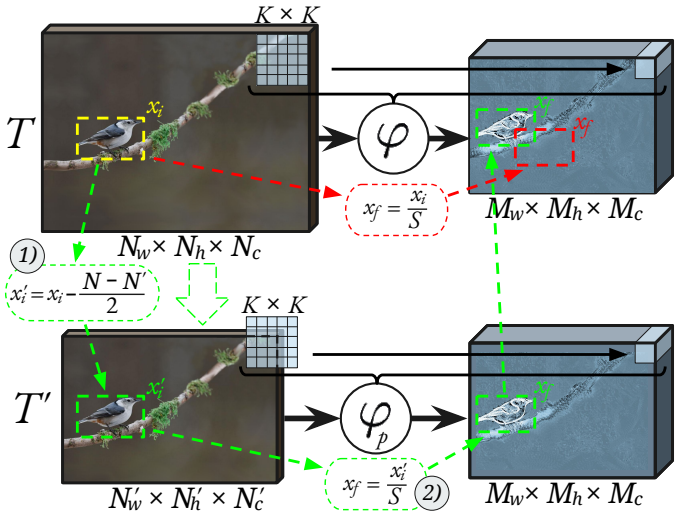

Fig. 2. Conversion between pixel coordinates and features coordinates using RoIAlign (colored red) and our proposal (colored green). Our crop-and-resize operator $\tilde{\kappa}$ provides correct coordinates for backbones without padding (φ), while the computation generated by RoiAlign [9] results in an off-centered bounding-box. Therefore, to obtain the correct coordinates, we first calculate the translation of the object when clipping the image (step `1)`), and then we transform those coordinates as RoiAlign would normally do (step `2)`). For instance, for a 1280×720 frame ($N' = 1193 \times 633$), an object centered in $x_i = (300, 380)$, and a backbone as in SiamMT ($K = 87$, $S = 8$, and without padding), with $\tilde{\kappa}$ we get a position in features of $x_f = (32, 42)$, while RoiAlign gives $x_f = (38, 48)$.

As an RoIPool successor, aiming for higher precision, RoIAlign [9] arises, which prevents misalignments by avoiding quantizations and computes the values of each bin by aggregating —maximum or average— 4 bilinearly interpolated sampling points. Following this approach, the $\tilde{\kappa}$ operator, introduced in SiamMT for the cropping and resizing of features, is an RoIAlign variant with only one sampling point per bin and a different region delimitation —see below. This allows $\tilde{\kappa}$ to maintain the inference speed while obtaining more precise and representative values than with RoIPool.

Features coordinates calculation. For the correct determination of the regions of interest, it is necessary to apply an additional modification to the features crop operation $\tilde{\kappa}$. In RoIPool and RoIAlign, the transformation between image coordinates x_i and features coordinates x_f is done by simply dividing the pixel coordinates by the backbone's global stride S. This, whilst it works when the feature extractors are architectures such as VGG [29], ResNet [30] or DarkNet [31], is incorrect if the backbone has no padding, as is the case for the AlexNet in [7] (Fig. 2, in red). Thus, in order to calculate the region coordinates for backbones without padding, it is necessary to apply transformations considering the *effective* size of the input tensor.

Let T be an input tensor of size N that produces an output of size M after passing through a fully-convolutional backbone φ without padding. We define N', the *effective* size of T with respect to φ, as the size of the minimum subtensor of T that produces an output of size M after passing through φ applying padding in all its operations —we denote this configuration of the backbone as φ_p. It is possible to prove that every tensor T would have an effective size of:

$$N' = \left\lceil \frac{N - K + 1}{S} \right\rceil \cdot S - (S - 1) \tag{1}$$

where K and S are the receptive field and the global stride of φ, respectively.

This implies that T generates an M-sized tensor after passing through φ, and T' (the N'-sized clipping of T) generates a tensor with exactly the same size M after passing through φ_p (Fig. 2, in green). By defining N' in this way, we can use T' as an intermediate step to transform coordinates extracted with a backbone without padding. Therefore, we first transfer the object coordinates (x_i) to its coordinates in T' (x_i') —as seen in Fig. 2, step `1)`—, and then we apply the standard RoIAlign transformation in order to obtain its correct position in the feature map (Fig. 2, step `2)`). Consequently, given some input coordinates in pixels x_i, their respective coordinates in features extracted with the fully-convolutional backbone φ without padding are calculated as follows, by concatenating transformations `1)` and `2)`:

$$x_f = \frac{1}{S} \cdot \left(x_i - \frac{N - N'}{2} \right) \tag{2}$$

Similarity operation. Finally, of particular interest is the reformulation of the features comparison operation applied at the end of the architecture. This is because, in SiamFC and its derived architectures, it is defined as a cross-correlation operation between the exemplar features and the search area features tensor. In the case of SiamMT, as it is necessary to cross-correlate pairs of tensors, this operation would have to be replicated throughout the batch size, which is computationally inefficient. As a solution, taking advantage of the properties of GPGPU architectures, we propose the use of a pairwise cross-correlation $\tilde{\star}$.

Pairwise cross-correlation (Fig. 3) is made possible by the properties of two-dimensional depthwise cross-correlation. The latter takes as inputs a three-dimensional tensor T_Y and a set of two-dimensional filters T_Z, and applies a different filter to each of the c channels of T_Y. The input of $\tilde{\star}$ will be a T_A tensor containing the features of $\mathcal{N}$ search areas and a T_E tensor containing the features of $\mathcal{N}$ exemplars. Therefore, by appropriately stacking the objects in T_A and T_E, it is possible to obtain the T_Y and T_Z tensors, which are the inputs to the depthwise cross-correlation operation. After this operation, an output with $\mathcal{N} \cdot c$ channels will be generated. Finally, this output is reshaped and aggregated for each object, which would correspond to the sum of the c channels of each filter in a standard correlation.

C. System training

Since SiamMT's architecture maintains the foundations of SiamFC, it is possible to reuse its weights through a network-based transfer learning procedure [8] for the tracking of multiple objects. This process consists in collecting all the parameters learned by a previously trained SiamFC network

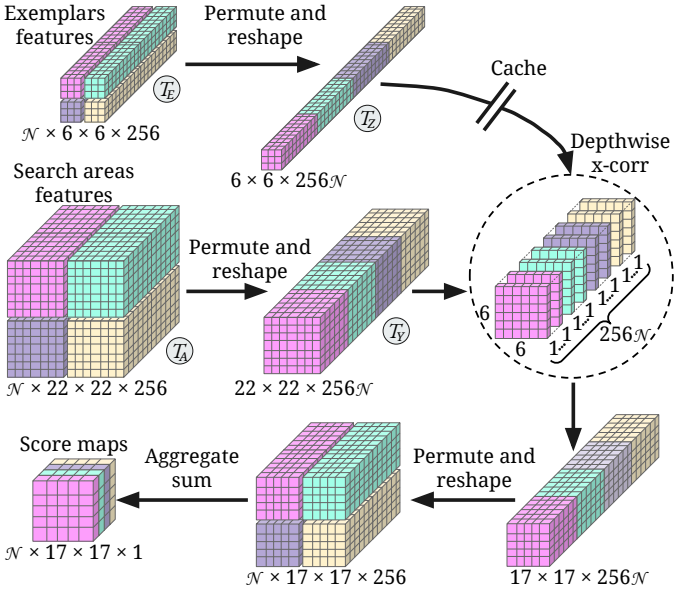

Fig. 3. Breakdown of the pairwise cross-correlation operator $\tilde{\star}$ in the SiamMT architecture for $\mathcal{N} = 4$ objects, each displayed on a different color. Notice how the transformed exemplar features can be cached during tracking to improve speed.

—backbone and similarity operation's final biases— and copying them into their respective SiamMT operations. This in itself provides good results, as can be seen in Section IV-B, hinting at the extensibility of SiamMT to other SiamFC-based architectures. However, due to the use of the crop and resize operation $\tilde{\kappa}$, it is convenient to fine-tune the last layers of the network. This fine-tuning is very similar to an end-to-end training of the network, but freezing the first 3 convolutional layers, and with a lower learning rate.

Regardless of whether the training is performed end-to-end or just fine-tuning, SiamMT's training error is calculated as the sigmoid cross-entropy loss

$$loss = \sum_{j=1}^{m} \sum_{i=1}^{n} \max(lg_{ij}, 0) - lg_{ij} \cdot gt_{ij} + \log(1 + e^{-|lg_{ij}|}) \quad (3)$$

where lg and gt are the $m \times n$-size matrices for the prediction score map and the ground truth, respectively. This ground truth is defined as in [7]. Also, to prevent the network from learning to detect objects rather than distinguishing among targets, negative examples are introduced during the training. Additionally, in order to gain tolerance against object scale changes, a random component is applied to the search area dimensions.

IV. Experiments

This section evaluates SiamMT under different scenarios. The experiments were conducted on a computer with an Intel Core i7-9700K, 16 GB of DDR4 RAM and an NVIDIA TITAN Xp. The chosen deep learning framework was TensorFlow.

TABLE I
Layer Structure of the Feature Extractor

Layer	Kernel	Filters	Stride	Channels	Size
input				×3	1280 × 720
conv1	11 × 11	96	2	×96	635 × 355
pool1	3 × 3		2	×96	317 × 177
conv2	5 × 5	128 (×2)	1	×256	313 × 173
pool2	3 × 3		2	×256	156 × 86
conv3	3 × 3	384	1	×384	154 × 84
conv4	3 × 3	192 (×2)	1	×384	152 × 82
conv5	3 × 3	128 (×2)	1	×256	150 × 80

A. Implementation details

Feature extractor. SiamMT uses the AlexNet-inspired [26] feature extractor described in Table I, as it offers a low computational cost. It is particularly important to note that no padding is introduced into the network, as this would result in it no longer being translation invariant [27]. It incorporates non-linear Leaky ReLU activation functions and batch normalization after each intermediate convolutional layer. Moreover, convolutions in layers 2, 4 and 5 are grouped, as it makes training faster and helps in learning better representations of the data [32].

Exemplar and search area sizes. For the formulation of the exemplar and search area sizes, a similar approach to SiamFC [7] is taken. However, since SiamMT calculates these regions over the frame features, the rescaling is performed to tensors of size 6×6 and 22×22, respectively, and without the need for additional context. Nevertheless, if the weights of the original SiamFC architecture are directly reused, the context value must be adjusted in order to reproduce its field of view.

Training process. The system was trained using the video database provided by the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [33]. The initial parameters are generated following the Kaiming method [34]. During training, Stochastic Gradient Descent (SGD) with a momentum of 0.9 is used to minimize the error function over 100 epochs, starting from a learning rate of 10^{-4} that is geometrically annealed to 10^{-7}. Each epoch consists of 65,000 pairs of frames —one for the exemplar and the other for the search area, containing the same object and spaced no more than 100 frames for the positive samples— arranged in size-8 batches. Lastly, in order to reduce the generalization error, we utilize data augmentation (translations, scale changes, rotations, color variations, motion blur and noise addition) and apply L2 regularization ($\lambda = 10^{-4}$) to the learned weights. The best model is selected based on the validation error of the last 60 epochs.

B. Tracking quality evaluation

SiamMT has to be evaluated within multiple-object environments. However, as tracking by detection is deeply rooted in the field of multi-object tracking, all their benchmarks focus on data association metrics [5], so they provide a set of detections

TABLE II
TRACKING QUALITY IN SINGLE-OBJECT BENCHMARKS

		SiamMT	SiamMT-W	SiamFC
OTB-15	Precision	73.52	72.37	77.12
	AUC	53.32	51.04	58.27
	fps	32.64	31.26	96.20
VOT-15	Accuracy	50.78	48.38	53.35
	Robustness	84.67	84.79	88.67
	fps	31.99	29.98	93.43

TABLE III
TRACKING QUALITY IN MULTI-OBJECT BENCHMARKS

		SiamMT	SiamMT-W	SiamFC
MOT-15	Accuracy	50.05	51.77	55.39
1125×679	Robustness	68.31	62.04	69.24
7.3 ob/im	fps	31.02	30.51	7.89
MOT-16	Accuracy	51.01	51.18	50.68
1718×986	Robustness	69.24	66.33	68.17
20.8 ob/im	fps	15.42	14.53	2.74
MOT-20	Accuracy	46.91	48.63	49.68
1620×1026	Robustness	76.27	71.37	71.17
149.7 ob/im	fps	6.11	5.56	0.53

in all the frames of the video as a basis and the challenge is to associate them correctly. Since SiamMT is not based on the association of detections across frames, by itself it is not MOTChallenge-compatible [35], and thus, it cannot be evaluated with its protocols and metrics.

With this in mind, in order to evaluate SiamMT and provide meaningful and comparable results, we have decided to address the problem by reporting individual tracking metrics within single and multiple object videos. This serves a twofold purpose: (i) it allows us to highlight the implications of adopting this new architecture, compared to the performance of the single object tracking network on which it is based; (ii) it provides a clear idea of how the network performs in scenarios with multiple objects, where interactions and occlusions are frequent. For the evaluation, we compare SiamMT with SiamFC [7] and SiamMT-W, i.e., SiamMT with its weights and parameters obtained from a previously trained SiamFC network through a transfer learning procedure [8].

Single-object-tracking benchmarks. The chosen single-object tracking benchmarks are those featured in [7] (OTB-2015 [36] and VOT-2015 [37]), whose results are shown in Table II. As can be seen, the precision and robustness scores for SiamMT are very similar to those of SiamFC, with a difference of less than 9%. This shows that the features generated by the feature resizing operator, while different from those obtained from a previously rescaled image, are still appropriate to discriminate between different objects. Moreover, SiamMT obtains slightly better scores than SiamMT-W, but the latter are still remarkably good considering that they have been obtained with no need for retraining or fine-tuning, highlighting the versatility of SiamMT's paradigm. In terms of speed, SiamMT and SiamMT-W have to process the entire frame, unlike SiamFC which considers a crop size of 255 px^2.

Multi-object-tracking benchmarks. To faithfully represent multiple-object-tracking scenarios, the public datasets for MOT-2015 [35], MOT-2016 [38] and MOT-2020 [39] were employed. As none of the evaluated trackers follow the tracking-by-detection paradigm, nor they consider detections during tracking, following the MOTChallenge protocol and reporting its metrics is not feasible. Therefore, we have opted for applying the VOTChallenge methodology [40], considering each object in each video as an individual sequence at the time of computing the metrics. All the results are listed in Table III, which also includes the average frame resolution and average number of objects per frame for each dataset.

As shown, SiamMT obtains accuracy results not less than 10% of SiamFC's, and displays an elevated robustness that surpasses SiamFC for MOT-2016 and MOT-2020. This emphasizes the benefits of SiamMT for multiple-object tracking since in this field it is more important to minimize identity switches than to accurately adjust the bounding-boxes. With respect to SiamMT-W, in spite of having a lower robustness than SiamMT, it achieves a slightly higher accuracy, which even surpasses SiamFC's for MOT-2016. These results suggest that the SiamMT paradigm can be straightforwardly applied to other SiamFC-based architectures with a minimal effort in training. As for the tracking speed, both SiamMT and SiamMT-W widely outperform SiamFC. For instance, in MOT-2020, with an average of 150 objects per frame, SiamMT is more than 10 times faster than SiamFC.

C. Tracking speed evaluation

We have designed two different test sets to represent the most typical multiple object tracking scenarios: the MT-VGA benchmark, composed of videos with a size of 640×480 pixels, and the MT-HD benchmark, with 1280×720-pixel videos.

To demonstrate the scalability of the proposed solution, the SiamMT architecture will be compared against the multiple instantiations of SiamFC [7] and SiamFC-MI. SiamFC-MI is a modified version of the SiamFC algorithm that is specially optimized for multiple instantiations since it stacks objects along its batch size to maximize GPU parallelization. However, it does not reuse features computations.

MT-VGA benchmark. The results for the MT-VGA benchmark are shown in Fig. 4a. For a single object, all three architectures offer speeds above 80 frames per second, with SiamFC being the fastest due to its simplicity, reaching 102 fps. SiamMT tracks one object at 92 fps and SiamFC-MI is the slowest with 81 fps, due to its batch management.

As more objects are added, the SiamFC and SiamFC-MI tracking speeds are reduced exponentially to just 11 and 16 fps for 10 objects, respectively. Meanwhile, the scalability of the SiamMT architecture becomes evident, dominating the rest of

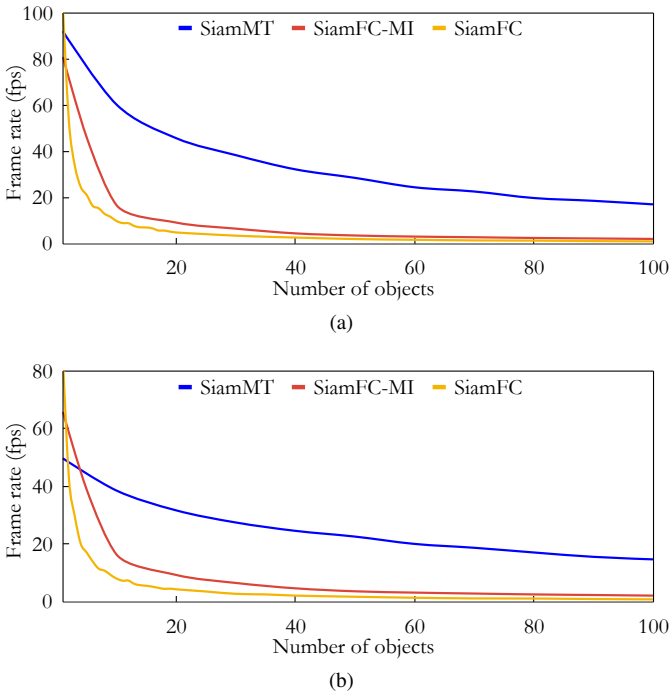

Fig. 4. Speed results for MT-VGA (a) and MT-HD (b) benchmarks with respect to the number of tracked objects.

the graph and allowing to track 10 objects at 60 fps. This is possible because SiamMT has to run the backbone only once per frame and because its final similarity operation exploits the properties of the depthwise cross-correlation to optimize pairwise tensor comparisons on GPU.

Thus, SiamMT scales almost linearly with the number of objects, being able to track 60 targets at 25 fps and reaching the benchmark limit —100 objects— with a speed of 17 fps. By contrast, even though SiamFC-MI offers higher speeds than SiamFC, its performance is far below SiamMT's, tracking 100 objects at only 3 fps.

MT-HD benchmark. The results for the MT-HD benchmark are shown in Fig. 4b. Since SiamMT has to perform the global features extraction of the 1280×720-pixel frames, it has a more noticeable overhead for a low number of objects, tracking 1 object at 50 frames per second while SiamFC and SiamFC-MI do it at 84 and 66 fps, respectively. The latter are virtually unaffected by the resolution of the frame, as they only process a crop of the image.

However, once above 5 objects, SiamMT outperforms the other architectures, and its computational cost remains almost constant due to its reuse of features computations. Thus, SiamMT is capable of tracking 40 objects at 25 fps and reaches a speed of 15 fps for 100 objects. Meanwhile, SiamFC and SiamFC-MI do not exceed 2 fps with 100 objects, evidencing the scalability offered by the SiamMT architecture.

V. Conclusions

We have proposed SiamMT, the first deep-learning-based arbitrary multi-object tracker. It applies individual visual tracking techniques to multiple objects in an efficient and scalable manner, on account of its global features extraction, its RoIAlign reformulation, and its optimized similarity operation. In experiments, SiamMT provides remarkable speeds, tracking 60 simultaneous objects in VGA video, and 40 objects in HD720 video, both at 25 fps. Additionally, it provides a tracking quality similar to SiamFC, and it is able to reuse the weights learned for SiamFC, suggesting that this paradigm can be straightforwardly applied to other SiamFC-based architectures.

Acknowledgment

This research was partially funded by the Spanish Ministerio de Ciencia e Innovación under grants TIN2017-84796-C2-1-R and RTI2018-097088-B-C32, and the Galician Consellería de Cultura, Educación e Universidades under grants ED431C 2018/29, ED431C 2017/69 and accreditation 2016-2019, ED431G/08. These grants are co-funded by the European Regional Development Fund (ERDF). Lorenzo Vaquero is supported by the Spanish Ministerio de Universidades under the FPU national plan (FPU18/03174).

References

[1] Y. Luo, D. Yin, A. Wang, and W. Wu, "Pedestrian tracking in surveillance video based on modified CNN," *Multimedia Tools Appl.*, vol. 77, no. 18, pp. 24 041–24 058, 2018.

[2] A. Rangesh and M. M. Trivedi, "No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars," *IEEE Trans. Intell. Veh.*, vol. 4, no. 4, pp. 588–599, 2019.

[3] L. A. Lim and H. Y. Keles, "Learning multi-scale features for foreground segmentation," *Pattern Anal. Appl.*, vol. 23, no. 3, pp. 1369–1380, 2020.

[4] W. Wang *et al.*, "Learning unsupervised video object segmentation through visual attention," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3064–3074.

[5] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. D. Reid, and S. Roth, "Tracking the trackers: An analysis of the state of the art in multiple object tracking," *CoRR*, vol. abs/1704.02781, 2017.

[6] M. Fernández-Sanjurjo, B. Bosquet, M. Mucientes, and V. Brea, "Real-time visual detection and tracking system for traffic monitoring," *Eng. Appl. Artif. Intell.*, vol. 85, pp. 410–420, 2019.

[7] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conf. Comput. Vis. (ECCV) Workshops*, 2016, pp. 850–865.

[8] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Int. Conf. Artif. Neural Netw. (ICANN)*, 2018, pp. 270–279.

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2980–2988.

[10] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 3701–3710.

[11] S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. R. Dick, and I. D. Reid, "Joint probabilistic matching using m-best solutions," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 136–145.

[12] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 6036–6046.

[13] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. Rhee, "Multi-class multi-object tracking using changing point detection," in *European Conf. Comput. Vis. (ECCV) Workshops*, 2016, pp. 68–83.

[14] Z. Zhou, J. Xing, M. Zhang, and W. Hu, "Online multi-target tracking with tensor-based high-order graph matching," in *Int. Conf. Pattern Recognit. (ICPR)*, 2018, pp. 1809–1814.

[15] K. Yoon, Y. Song, and M. Jeon, "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views," *IET Image Process.*, vol. 12, no. 7, pp. 1175–1184, 2018.

[16] B. Cuan, K. Idrissi, and C. Garcia, "Deep siamese network for multiple object tracking," in *Int. Workshop Multimedia Signal Process. (MMSP)*, 2018, pp. 1–6.

[17] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 941–951.

[18] Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," *CoRR*, vol. abs/1909.12605, 2019.

[19] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 2544–2550.

[20] H. K. Galoogahi, T. Sim, and S. Lucey, "Multi-channel correlation filters," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2013, pp. 3072–3079.

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

[22] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 1781–1789.

[23] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8971–8980.

[24] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 1328–1338.

[25] M. Kristan *et al.*, "The seventh visual object tracking VOT2019 challenge results," in *IEEE Int. Conf. Comput. Vis. (ICCV) Workshops*, 2019, in press.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1106–1114.

[27] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 4282–4291.

[28] R. B. Girshick, "Fast R-CNN," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1440–1448.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. Learn. Repr. (ICLR)*, 2015, pp. 1–14.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[31] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 6517–6525.

[32] Y. Ioannou, D. P. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving CNN efficiency with hierarchical filter groups," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 5977–5986.

[33] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1026–1034.

[35] L. Leal-Taixé, A. Milan, I. D. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," *CoRR*, vol. abs/1504.01942, 2015.

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

[37] M. Kristan *et al.*, "The visual object tracking VOT2015 challenge results," in *IEEE Int. Conf. Comput. Vis. (ICCV) Workshops*, 2015, pp. 564–586.

[38] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *CoRR*, vol. abs/1603.00831, 2016.

[39] P. Dendorfer *et al.*, "MOT20: A benchmark for multi object tracking in crowded scenes," *CoRR*, vol. abs/2003.09003, 2020.

[40] M. Kristan *et al.*, "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2137–2155, 2016.