CapsuleRRT: Relationships-aware Regression Tracking via Capsules

Ding Ma, Xiangqian Wu
School of Computer Science and Technology, Harbin Institute of Technology
madingcs@hit.edu.cn, xqwu@hit.edu.cn✉

Abstract

Regression tracking has gained more and more attention thanks to its easy-to-implement characteristics, while existing regression trackers rarely consider the relationships between the object parts and the complete object. This would ultimately result in drift from the target object when missing some parts of the target object. Recently, Capsule Network (CapsNet) has shown promising results for image classification benefits from its part-object relationships mechanism, while CapsNet is known for its high computational demand even when carrying out simple tasks. Therefore, a primitive adaptation of CapsNet to regression tracking does not make sense, since this will seriously affect speed of a tracker. To solve these problems, we first explore the spatial-temporal relationships endowed by the CapsNet for regression tracking. The entire regression framework, dubbed CapsuleRRT, consists of three parts. One is S-Caps, which captures the spatial relationships between the parts and the object. Meanwhile, a T-Caps module is designed to exploit the temporal relationships within the target. The response of the target is obtained by STCaps Learning. Further, a prior-guided capsule routing algorithm is proposed to generate more accurate capsule assignments for subsequent frames. Apart from this, the heavy computation burden in CapsNet is addressed with a knowledge distillation pose matrix compression strategy that exploits more tight and discriminative representation with few samples. Extensive experimental results show that CapsuleRRT performs favorably against state-of-the-art methods in terms of accuracy and speed.

1. Introduction

Regression tracking can be broadly divided into two categories: discriminative correlation filters (DCFs) based trackers [22, 42, 8] and deep regression networks (DRNs) based trackers [7, 2, 9, 33, 27]. DCFs trackers exploit the property of the circulant matrix and optimize the correlation filters with a system of linear functions. Not only that, DCFs trackers achieve fast speed by taking advantage of the correlation computed through the Fourier domain, but also perform favorably on several popular tracking benchmarks [41, 24, 11]. Unfortunately, the main drawback of DCFs trackers is that the DCFs trackers take few advantages of end-to-end learning. On the contrary, DRNs trackers [7, 2, 9, 33, 27] have paid more attention due to it has the potential to take full advantage of end-to-end learning. Most DRNs trackers share a similar mechanism: learn a mapping from a sampling area of the target objects to soft labels derived from a Gaussian function. Nevertheless, such a mechanism does not take into account the relationships between the object parts and the complete object, thus giving rise to drift. As shown in Figure 1, low response values are assigned to some parts of the object, thus resulting in drift. Further, some DRNs trackers update with such noisy samples, which will result in tracking failure.

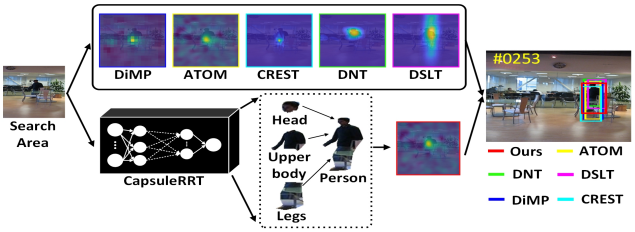


Figure 1: Visual comparisons of the response map in existing deep regression networks on the $Human2$ sequence. As our CapsuleRRT takes the relationships of part-object into account, a relatively correct response can be obtained even in the case of heavy occlusion.

Recently, Hinton *et al*. [14] proposed the idea of CapsNet, which models the spatial relationships with capsule architecture. However, it seems that directly apply the original CapsNet to deep regression tracking may not work, which can be explained from three aspects. Firstly, the communication between two-level capsules is heavily relied on the pose matrices of two-level capsules. Specifically, the vote matrix of low-level capsules for the adjacently high-level capsules is computed by multiplying the pose matrix and transformation matrix. A large dimensional pose matrix will introduce much unnecessary noise. This is well verified by noisy capsules produced by CapsNet within the solid boxes of Figure 2, which will cause the weak discriminant

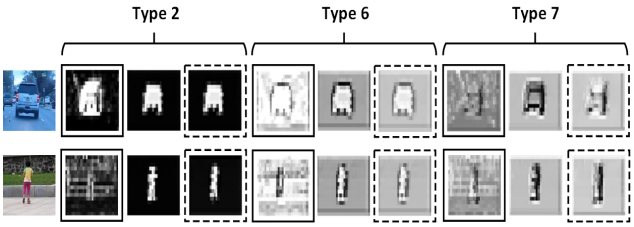


Figure 2: Visual comparisons of PM with different size on the $BlurCar1$ and $Girl2$ sequences. The one marked by the solid box, no box, and dotted box indicates the PM with 32-D, 16-D, and 8-D $w/$ KDPMC strategy. Capsules of the ConvCaps1 layer are displayed, the same below.

to separate the target object from noisy background. Secondly, the original CapsNet require a high computational demand even if it is carried out the digital image classification task. Hence, CapsNet need to simplifying key algorithmic components to meet the high-speed demand of regression tracking. Thirdly, compared with static image classification, tracking tasks need to consider the temporal relationships, challenging the original routing algorithm.

To fully take advantage of relationships learned by CapsNet and solve the above issues, we propose, a spatial-temporal relationships-aware capsule architecture for regression tracking, where three components are designed to implement CapsNet for effective exploring part-object relationships. Specifically, we design a spatial capsules module (S-Caps), which takes the spatial part-object relationships into account. In such way, the relevant parts will be clustered together to form a complete object through the spatial domain. To capture temporal relationships across the frames, we design a temporal capsules module (T-Caps) to generate a set of temporal capsules and then pass these capsules through the STCaps Learning module, which allows the temporal capsules to condition to spatial capsules.

Besides, there should be strongly correlated types of capsules between the first frame and the current frame. Specifically, the first frame capsules are more reliable, which are derived from the only labeled sample (i.e., initial state of the object) during inference stage. Therefore, we design a prior-guided capsule routing algorithm (PGR), where the prior correlations computed from the first frame, are used to estimate the capsule assignments for the next frame. In such a way, capsule assignments can be achieved to explore the relatively right part-object relationships.

Until now, the heavy computational burden inherited from CapsNet makes it difficult to achieve the purpose of fast-tracking. We address this issue with a knowledge distillation compression strategy to efficiently reduce the parameters of those three modules. In detail, our compression strategy contains three steps: compressing each pose matrix (PM) to obtain a compact pose matrix (PM-short), aligning PM-short to PM by an aligned pose matrix (AM), and merging the added AM into PM-short. Since there are few parameters to estimate, we can get a good estimation with a small amount samples, which also meet the demand of tracking task (i.e., only the initial state of the target is given). As shown in the dotted boxes of Figure 2, the compressed capsules are distinctive enough to separate the object from the background. Besides, our capsule architecture has much fewer parameters and noisy assignments.

We make the following contributions in this work:

I, We incorporate a new property, i.e., spatial-temporal relationships, in regression tracking, which is implemented by CapsNet. To the best of our knowledge, this is the first attempt to apply CapsNet for regression tracking.

II, We propose a S-Caps module for capturing spatial relationships, and a T-Caps module for encoding temporal relationships. These two modules are jointly clustering parts to form the target object within STCaps Learning.

III, We design a novel capsule routing algorithm, named prior-guided routing algorithm (PGR), which maintains capsule information from the first frame to guide for capsules routing through subsequent frames.

IV, We propose a knowledge distillation pose matrix compression strategy (KDPMC) to adapt the proposed architecture to fast-tracking, aiming to select more tight and discriminative pose matrix representation.

V, We compare our approach with the state-of-the-art methods on seven datasets, which consistently show the superiority of our algorithm.

The rest of the paper is organized as follows. We first review the related works in Section 2. Then, we present the detailed configuration of the proposed framework and provide the optimization process in Section 3. After that, we provide some detailed analysis in Section 4. We further illustrate the implementation details and experimental results on popular tracking benchmarks in Section 5. Finally, conclusions are drawn in Section 6.

2. Related Work

2.1. Regression Tracking

Regression methods directly regress a response map from a regularly dense sampling. One representative category is based on Discriminative Correlation Filters (DCFs), which regress all the circularly shifted versions of an input image to a response map for target localization. By computing the ridge regression in the Fourier domain, DCFs trackers showing attractive efficiency [4, 35, 16]. Besides DCFs, with the recent delightful progress of deep learning, Deep regression networks (DRNs) have gained significant popularity in visual tracking. Among these approaches, a CNN kernel is learned to convolve with the dense sampling features for response generation. FCNT [39], one of the pioneering work in the development of DRNs, which de-

signs two fully convolutional subnetworks to capture high-level and low-level features of the target jointly. Similarly, DNT [6] is proposed to gather local and global information through dual deep networks. CREST [33] fuses the outputs of the baseline and another two residuals to estimate the location of the target object. Besides, DSLT [27] proposes a shrinkage loss and an ensemble strategy to improve the performance of the DRNs trackers. To ensure a real-time online process, ATOM [7] employs a conjugate-gradient-based strategy in the deep learning framework. DiMP [2] introduces a target model prediction network and trains the whole model in an end-to-end manner. Recently, PrDiMP [9] proposes a formulation for learning to predict the conditional probability density of the target, which is capable of modeling label noise stemming from ambiguities in the task. Despite the recent progress, DRNs trackers are rarely paid attention to the relationships within the target object, which will result in drift by missing parts of the object.

2.2. CapsNet

CapsNet [14], which models the spatial relationships of the target with capsules and dynamic routing. Given its advances, researchers have introduced some applications of CapsNet. SegCaps [20] embeds the capsules into U-Net architecture for segmenting pathological lungs from low dose CT scans. CapsuleVOS [10] designs a novel attention-based EM routing algorithm to condition capsules for video object segmentation. Recently, TSPOANet [26] proposes a two-stream part-object assignment network that routes the low-level capsules to high-level capsules in a local region to reduce the parameters of CapsNet. Nevertheless, such a simple strategy ignores the distributions of the parameters in transformation weights, which will result in inaccurate assignments during the routing process. Compared with it, the primary distinction of this work is that our compressed capsule architecture is not only to reduce the parameters of CapsNet but also to avoid the inaccurate assignment.

2.3. Network Compression

Recently, few-samples knowledge distillation based network compression [1, 23] has gained lots of attention, as the student-nets can achieve competitive accuracy even under the few-sample setting. Similarly, tracking methods need to be initialized have a high response to the specified target with few samples given in the initial frame. Our method is also heavily motivated by few-samples knowledge distillation-based network compression [23], where we aim to compress the pose matrix of each capsule to reduce the parameters without sacrificing accuracy.

3. Proposed Method

Figure 3 shows the overall architecture of the proposed tracking framework. At the beginning, the representations

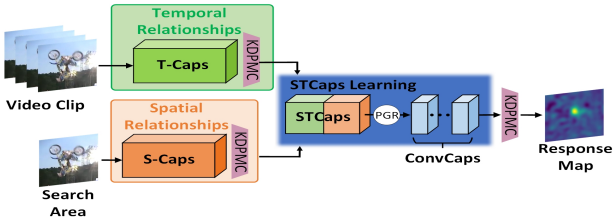


Figure 3: CapsuleRRT architecture. Given the video clip and the search area, we explore the spatial-temporal relationships via S-Caps, T-Caps, and STCaps Learning.

of the target are fed into the proposed S-Caps module to exploit spatial relationships. Meanwhile, we extract temporal relationships through a T-Caps module by given a video clip. The optimal spatial-temporal relationships are learned by STCaps Learning. Benefit from the proposed PGR algorithm, which guides the capsule assignments for subsequent frames more accurate. In the end, a KDPMC strategy helps to reduce the computational cost while preserving more discriminative capsule assignments.

3.1. S-Caps

S-Caps aims to extract spatial relationships by given a search area. According to Figure 4a, we use one convolutional layer to integrate the features of *conv4* and *conv5* of VGG-16 [32] to obtain the representations $36 \times 36 \times 128$ of the given search area. Followed by these representations, we generate capsules by emerging two branches. Specifically, we feed the feature maps into a SpatialCaps layer in which each capsule is constructed by a pose matrix and an activation value. The pose matrix encodes the pose properties. The activation value indicates the existence of the entity. The details are shown as follows.

Channel Reduction. The feature maps with the dimension of $36 \times 36 \times 128$ are downsampled with one convolutional layer to $36 \times 36 \times 16$ for reducing the number of channels.

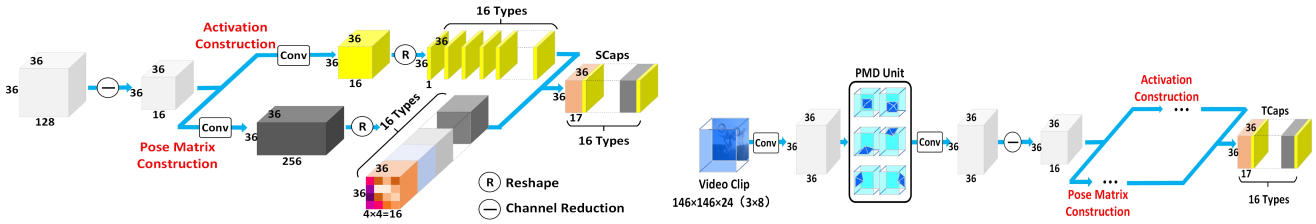
Pose Matrix. Through one convolutional layer, the $36 \times 36 \times 16$ feature maps are enriched to $36 \times 36 \times 256$. Then, the pose matrices are generated by reshaping the obtained feature maps to $36 \times 36 \times 16 \times 16$.

Activation Value. The $36 \times 36 \times 16$ feature maps are transformed to $36 \times 36 \times 16$ by a convolutional layer to match the number of types of capsules. The activation values can be represented by $36 \times 36 \times 16 \times 1$.

Capsules Generation. In the end, the 16 types of SCaps (i.e., $36 \times 36 \times 16 \times 17$) are generated by concatenating the pose matrices and corresponding activation values.

3.2. T-Caps

We propose a T-Caps module to strengthen S-Caps with consecutive temporal relationships. The architecture of the proposed T-Cpas is shown in Figure 4b. Firstly, 8 RGB



(a) (b)

Figure 4: (a) and (b) show the detailed architecture of S-Caps and T-Caps, respectively.

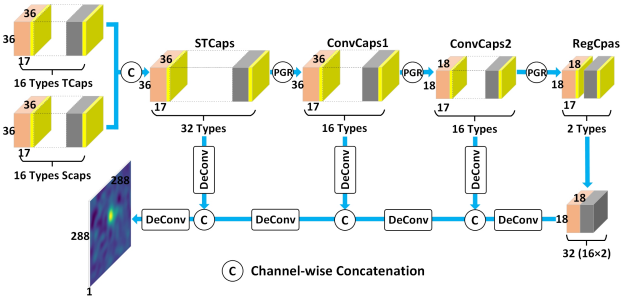


Figure 5: Detailed architecture of STCaps Learning.

search areas ($146 \times 146 \times 3$) are concatenated along the channel dimension to $146 \times 146 \times 24$. The obtained features are then downsampled to $36 \times 36 \times 36$ through 2 convolutional layers. After that, we use a Parallel Multi-Dimensional (PMD) unit [34] to model contextual dependencies. The PMD unit contains 6 direction Multi-Dimensional LSTM, which is shown in Figure 4b. Followed by PMD, we generate TCaps ($36 \times 36 \times 16 \times 17$) through a TemporalCaps layer, which is similar to SpatialCaps. The details can be found in Figure 4a.

3.3. STCaps Learning

By leveraging the spatial-temporal relationships derived from S-Caps and T-Caps, we obtain the response map of the specific target by a STCaps Learning module. As shown in Figure 5, those capsules obtained by S-Caps and T-Caps are concatenated along the type axis, forming the STCaps (i.e., $36 \times 36 \times 32 \times 17$). Afterward, there are two Convolutional Capsules (i.e., ConvCaps1 and ConvCaps2) layers, which share the same architecture. In the end, a RegCaps layer is designed to classify the target and background. Here, we take ConvCaps1 for example to illustrate the details.

Reshape capsules. The capsules are first reshaped to $1296 \times 32 \times 17$, i.e., the pose matrices and activation values are $1296 \times 32 \times 16$ and $1296 \times 32 \times 1$, respectively.

Votes between two-layer capsules. The vote V_{ij} of capsule i for capsule j is obtained by:

$$V_{ij} = M_i W_{ij} \tag{1}$$

where $M_i \in R^{4\times4}$ is the pose matrix of the capsule i in one layer, the trainable transformation matrix $W_{ij} \in R^{4\times4}$ acts as a communicator between the capsule i and j which is a capsule in the layer above. The votes are $1296\times32\times16\times16$, where 1296, 32, 16, and 16 are the number of capsules of one type, type number of low-level and high-level capsules, and the dimension of vote matrix, respectively.

Route capsules. The pose matrices and the activation values of low-level capsules are input into the iterative routing algorithm, which will calculate the pose matrices (i.e., $36 \times 36 \times 16 \times 16$) and activation values (i.e., $36 \times 36 \times 16 \times 1$), respectively. The high-level capsules (i.e., $36\times36\times16\times17$) are obtained by concatenating operation.

ConvCaps2 has a similar architecture with ConvCaps1. The output of ConvCaps2 is $18 \times 18 \times 16 \times 17$. At the end of ConvCaps2, we design a RegCaps layer to classify the target and the background. The architecture of RegCaps is also similar to ConvCap1, except that the type is set to 2 to correspond to the target and the background, respectively.

To estimate the response of the target object, we utilize the representation of the RegCaps layer's pose matrices. As shown in Figure 5, we upscaled the pose matrix of RegCaps to $36 \times 36 \times 64$. These features are then upscaled through a series of deconvolutions that result in a $288 \times 288 \times 1$ response map with the same size as the search area.

3.4. Prior-Guided Capsule Routing

To extend the original routing algorithm [14] to streaming video, we resort to the initial capsule similarities for more robust capsule routing. Suppose f_I and f_{JO} are the features of $type_I$ capsules of the current frame and $type_J$ of the first frame, the guided coefficient is encoded as:

$$c_{JO} = 1 - I(f_I, f_{JO}) \tag{2}$$

$$I(f_I, f_{JO}) = \frac{f_I \cdot f_{JO}}{||f_I||||f_{JO}||} \tag{3}$$

where $\cdot$ indicates the inner product operation. This prior-guided information is embedded into the EM routing by:

$$R_{ij} = R_{ij} + R_{ij} * c_{JO} \tag{4}$$

where R_{ij} is the capsule assignment probability. i and j are two capsules belonging to $type_I$ and $type_J$ in current

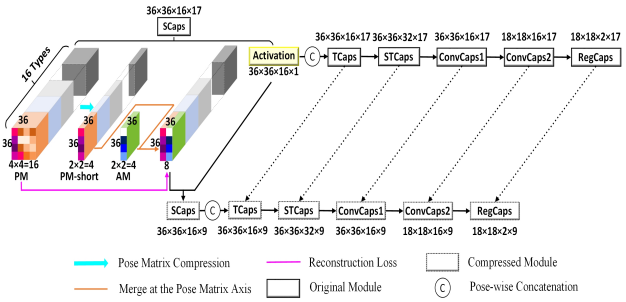


Figure 6: The detailed architecture through KDPMC.

frame, respectively. There are five steps in our PGR, and the details of these steps are illustrated as follows.

Initialization. The assignment probability R_{ij} is initialized with uniform distribution. We initialize the c_{JO} to be zero.

M-procedure. By given the activation a_i, votes V, and the current R_{ij}, M-procedure computes an updated Gaussian model (μ, σ) and the activation a_j of high-level capsule j.

E-procedure. Given the (μ, σ) and the activation a_j, E-procedure determines the assignment probability R_{ij} of each low-level capsule to a high-level capsule.

Prior-guided information embedding. The prior-guided information is embedded into the assignment probability to gather higher similar capsules which are higher correlated to the most reliable state of the target object.

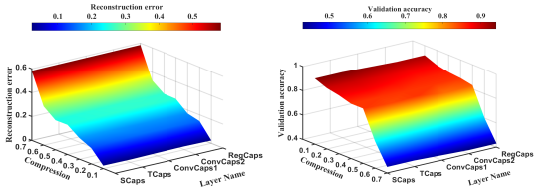
Prior-guided information calculation. We calculate the prior-guided information of different types of capsules.

The designed R_{ij} is able to activate those high-familiarity capsules while alleviating confusing capsule assignments. As well, the training procedure will be speeded up with the guidance of prior-guided information, and we have verified this in the section of experiments.

3.5. Knowledge Distillation Pose Matrix Compression Strategy

To reduce the computation burden of CapsNet [14], we propose a knowledge distillation pose matrix compression strategy (KDPMC). We first compress PM and obtain PM-short. Secondly, we add an AM at the end of each PM-short. Then, we forward a few training data to both the PM and PM-short, and align the output of PM-short to PM by estimating the parameters of AM. We obtain the estimation with a small number of samples by taking advantages of few parameters of AM. By taking SCaps as an example, we illustrate the details of each step as follows (see Figure 6).

Compress the Pose Matrix. In the SCaps, the pose matrix is denoted as $36 \times 36 \times 16 \times 16$, i.e., each type of capsule is represented as $36 \times 36 \times 16(4 \times 4)$. We obtain the PM-short $36 \times 36 \times 4(2 \times 2)$ by PCA. PM-short inherits core representations from PM, so adding a small group of capsules is enough to preserve the representative ability.



(a) (b)

Figure 7: Plots of reconstruction error (a) and validation accuracy (b).

Type-level Alignment. Suppose O^{ori} is the type-level output of the original pose matrix PM. Given one type of the PM and the corresponding PM-short, we add an AM at the end of each type capsule. The parameters of AM are initialized with the mean of PM. The parameters of AM can be estimated with a minimum mapping Map [12]. The total loss of SCaps can be minimized the reconstruction loss as:

$$\mathcal{L}_{align}^{SCaps} = \arg\min \sum_{j=1}^{M} \sum_{i=1}^{N} Map(O_{ij}^{as} - O_{ij}^{ori}) \quad (5)$$

where M is the number of types of capsules in SCaps, and N is the number of training samples. The output of AM and PM-short is indicated as O^{as}.

Merge the PM-short and AM. The compressed PM ($64 \times 64 \times 16 \times 8$) is obtained by concatenating PM-short and AM along the pose matrix axis.

Similarly, we use the same steps above to compress the transformation matrix of each type capsules. In Figure 7a, the reconstruction error does not increase much until 50% compression (i.e., 8-dimensional pose matrix). In Figure 7b, the model is relatively unaffected for 50% compression. In light of these, we designed the compressed capsule architecture in Figure 6. We compress the decoder network through [23] to match the compressed dimensions.

3.6. Optimization and Tracking

For offline training, we have ground truth soft label y, and the learned feature x. We use the regression loss $\mathcal{L}_{reg}$ [27] for learning the whole model. For online initialization, the model is trained using an objective function which is the sum of three loss: regression loss, aligned loss of capsule layers, and aligned loss of decoder. The total objective function to be minimized by:

$$\mathcal{L}_{total} = \mathcal{L}_{reg} + \sum \mathcal{L}_{aglin}^{Caps*} + \mathcal{L}_{aglin}^{decoder} \quad (6)$$

where $Caps*$ indicates the five capsule layers.

In the offline training stage, we train the whole model with $\mathcal{L}_{reg}$. In the initialization stage, the total sample number is set to 100 to adapt the parameters of the compressed

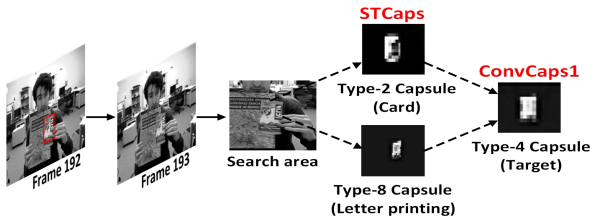


Figure 8: Relationships on the $ClifBar$ sequence.

model to the specific target. The compressed model is optimized by the objective function in 6. During tracking stage, we crop a search area centered at the estimated location in the previous frame and feed into S-Caps. For the T-Caps, we use the method in [31] to interpolate the frames until the 8^{th} frame. The location of the maximum value of response map indicates the position of target. We carry out scale estimation using the scale pyramid strategy as in [27]. We only update the compressed model per 20 frames with $\mathcal{L}_{reg}$.

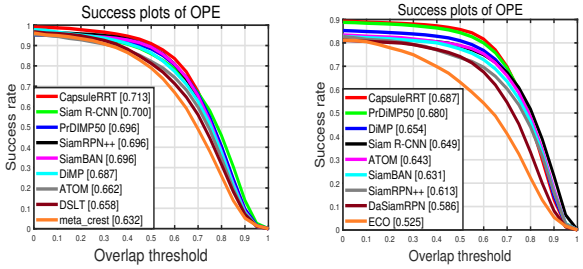
4. Detailed Analysis

In visual tracking, the relationships are derived from that two parts will be clustered together to form a whole target object. As in Figure 8, type-2 and type-8 capsules from the STCaps focus on two parts, i.e., card and letter printing. In the higher-level, type-4 capsules encode the whole object. Through voting, type-2 capsules cooperate with type-8 capsules to form a complete object at the higher type-4 capsules. This shows that our method can capture the relationships among the target, making up for the lack of relationships in DRNs trackers.

5. Experiments

5.1. Implementation Details

Our tracker is implemented in Pytorch, and runs on a PC with a 1080ti GPU. The feature extractor is initialized with the ImageNet weights. The input search region is resized to 288×288. We utilize the training split of LaSOT [11], TrackingNet [29], GOT-10k [15], and COCO [25] for offline training. The model is trained for 100 epochs with 1000 iterations per epoch and 36 image pairs per batch. The ADAM optimizer [18] is used with an initial learning rate of 0.01, and set a decay factor 0.2 per 10 epochs. During online update, we decrease the learning rates to 2e-5, and the training samples are collected by the maximum response value > 0.5. A buffer for T-Caps maintains the last 8 frame's results. At the beginning, we perform data augmentation by constructing 100 training samples. We use these samples to optimize the parameters of the whole model for 50 iterations. The training label is generated by a Gaussian function with a kernel width proportional (0.1). The routing iteration is set to 3. The kernel of PMD is set to 3.



(a) (b)

Figure 9: Success plots on the OTB100 (a) and UAV123 (b). DiMP [2] is the ResNet-50 version, the same below.

Table 1: Comparison on the GOT-10k in terms of Average overlap (AO) and Success rate (SR). The best two results are highlighted in red and blue, the same below.

	Ours	Siam R-CNN [36]	PrDiMP50 [9]	DIMP [2]	DCFST [46]	ATOM [7]	KYS [3]	SPM [38]	ECO [8]
$SR_{0.50}$	0.748	0.728	0.738	0.717	0.716	0.634	0.751	0.593	0.309
$SR_{0.75}$	0.524	0.597	0.543	0.492	0.463	0.402	0.515	0.359	0.111
AO	0.656	0.649	0.634	0.611	0.610	0.556	0.636	0.513	0.316

Table 2: Comparison on the VOT2019 in terms of Robustness (R), Accuracy (A) and Expected Average Overlap (EAO).

	Ours	DiMP [2]	DCFST [46]	SiamBAN [5]	SiamDW [45]	ARTCS [19]	ATOM [7]	SiamMask [40]
EAO	0.392	0.379	0.361	0.327	0.299	0.294	0.292	0.287
A	0.610	0.594	0.589	0.602	0.600	0.602	0.603	0.594
R	0.265	0.278	0.376	0.396	0.467	0.482	0.411	0.461

5.2. Comparison with the State-of-the-art

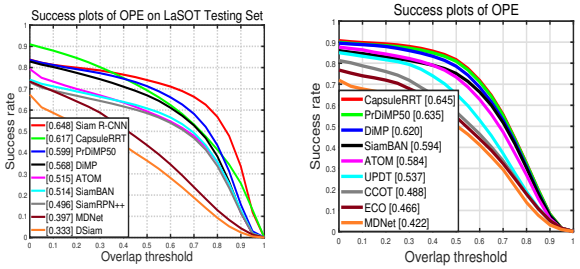
OTB100 [41]. OTB100 [41] is a widely evaluated dataset contains 100 sequences, Figure 9a shows the success plot on OTB100. Compared with the popular regression trackers including PrDiMP50 [9], DiMP [2], ATOM [7], our method outperforms them by 1.7%, 2.6%, and 5.1% in terms of AUC score, respectively, which demonstrates the effectiveness of the part-object relationships in the CapsuleRRT.

UAV123 [28]. UAV123 dataset [28] is a recently released dataset which contains 123 tracking targets. Figure 9b illustrates the results of eight state-of-the-art trackers. It is delightful that our tracker outperforms all the compared methods including non-real-time trackers in the success plot. It indicates that our CapsuleRRT has better generalization capability to track the target captured by UAV platform.

GOT-10k [15]. GOT-10k dataset [15] is a recent large-scale dataset consisting of 10K for training and 180 for testing. We follow the defined protocol [15] and only train on the specified GOT10k training set for this experiment, while keeping all other settings the same. The results are shown in

Table 3: Comparison on the TrackingNet test set in terms of Precision, Normalized Precision, and Success.

	Ours	Siam R-CNN [36]	PrDiMP50 [9]	MAML [37]	ROAM [43]	SiamAttn [44]	DiMP [2]	ATOM [7]	KYS [3]
Precision	0.726	0.800	0.704	0.725	0.623	-	0.687	0.648	0.688
Norm. Prec.	0.834	0.854	0.816	0.822	0.754	0.817	0.801	0.771	0.800
Success	0.774	0.812	0.758	0.757	0.670	0.752	0.740	0.703	0.740



(a) (b)

Figure 10: Success plots on the LaSOT (a) and NFS (b) datasets.

Table 1, obtained through a public server, ranked in terms of SR and AO. Compared with the recent regression trackers, our tracker achieves competitive results in three metrics. Our tracker improves the AO by 0.7 points over the non-real-time tracker Siam R-CNN, while outperforming Siam R-CNN [36] by 2 points in terms of $SR_{0.5}$.

VOT2019 [19]. VOT2019 [19] is a recently released challenging benchmark. As shown in Table 2, compared with the DiMP [2] and ATOM [7] from the regression category, our tracker ranks first in terms of EAO, A, and R.

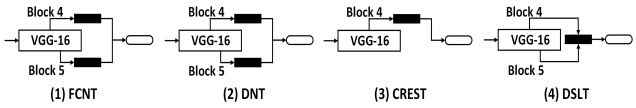
TrackingNet [29]. TrackingNet [29] is a recently introduced dataset consisting of real-world videos sampled from Youtube. The tracker is evaluated using an online evaluation server on a test set of 511 videos. Table 3 shows the results in terms of precision, normalized precision, and success. Our method surpasses the recent regression-based methods such as PrDiMP50 [9], DiMP [2], and ATOM [7] with a relative gain of 1.6%, 3.4%, and 7.1% in success, respectively. Among all the compared methods, only Siam R-CNN [36] outperforms ours, which runs only at 4.7 FPS.

LaSOT [11]. LaSOT dataset [11] is a very large-scale dataset consisting of 1400 sequences with 70 categories. Our approach is evaluated on the test set of 280 videos. Except for the MDNet [30] and DSiam [13], we add the recent Siam R-CNN [36], PrDiMP50 [9], DiMP [2], ATOM [7], SiamBAN [5], and SiamRPN++ [21] for comparison. As shown in 10a, our method achieves an AUC score of 0.617, surpassing all the real-time trackers. Especially, our CapsuleRRT outperforms the previous best regression method (i.e., PrDiMP50) by a large margin of 1.8% AUC score.

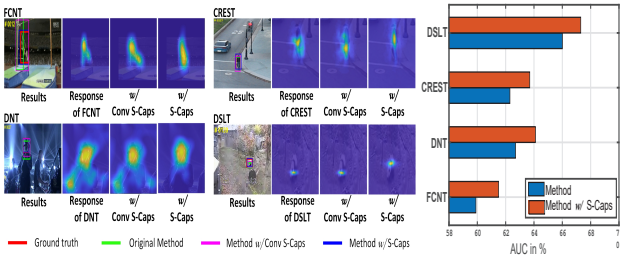
NFS [17]. NFS dataset [17] includes 100 videos that are captured from real-world scenarios. We evaluate our tracker on the 30 FPS version of NFS. Figure 10b shows the success plot. CapsuleRRT achieves a substantial improvement over the previous state-of-the-art on this dataset.

Table 4: Performance evaluations for the ablation analyses on the AUC score of success plot through OTB100 [41] and LaSOT [11]. Each variant followed by the STCaps Learning module, expect for (a) where the input of STCaps Learning module is only the SCaps. The communication between capsules is the original EM routing algorithm [14] in (1,2).

	S-Caps	T-Caps	STCaps Learning	PGR	KDPMC	OTB100 [41]	LaSOT [11]
(1)	✔		✔			0.678	0.535
(2)	✔	✔	✔			0.703	0.599
(3)	✔	✔	✔	✔		0.718	0.625
(4)	✔	✔	✔	✔	✔	0.713	0.617



(a)



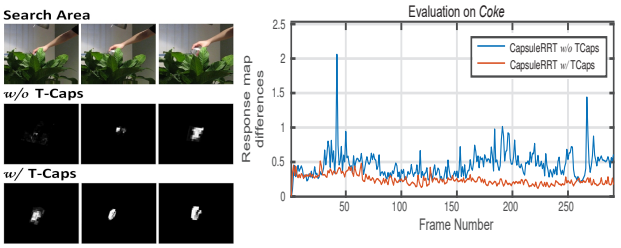
(b) (c)

Figure 11: (a) The simply structure of compared trackers, we replace the black blocks with our S-Caps and STCaps Learning. (b) Visual comparisons of each tracker on the *Jump*, *Huamn6*, *Shaking*, and *Panda* sequences. (c) Performance gains when replacing convolutional architecture with the proposed capsule architecture.

5.3. Ablation analysis

In this section, we verify the effectiveness of our framework through OTB100 [41] (i.e., short-term) and LaSOT [11] (i.e., long-term) datasets.

S-Caps. To show the robustness of S-Caps, we design a modified version, i.e., "Backbone + S-Caps + STCaps Learning". The performance is given in Table 4(1), which indicates that "Backbone + S-Caps + STCaps Learning" achieves competitive results on both of the datasets. This evidently proves that the spatial relationships play an important role in the DRNs. To further verify the necessity of spatial relationships, we replace the convolutional regression



(a) (b)

Figure 12: (a) Visual comparisons for the T-Caps on the *Coke* sequence. (b) Response map differences of the *Coke* sequence.

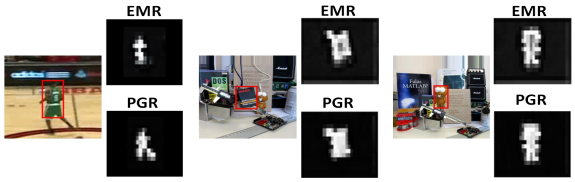


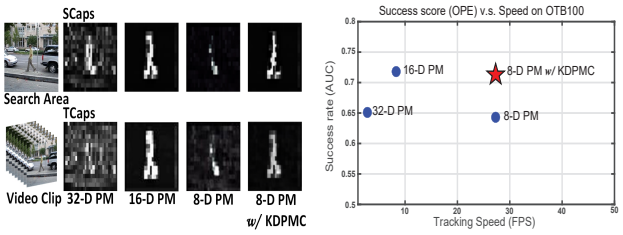
Figure 13: Visual comparisons for the EM routing (EMR) and PGR on the *Basketball*, *Box*, and *Lemming* sequences.

blocks of FCNT [39], DNT [6], CREST [33], and DSLT [27] with S-Caps and STCaps Learning (see Figure 11). Thanks to the relationships from S-Caps and STCaps, these trackers can generate the response map more precisely.

T-Caps. To illustrate the effectiveness of T-Caps, we compare it with a modified version that is obtained by adding T-Caps. The detailed comparisons for these two versions are shown in the of Table 4(1,2). It is obvious in Table 4(2) that T-Caps increases AUC scores by 2.5% and 6.4% points in OTB100 and LaSOT datasets, respectively. As shown in Figure 12a, we can find that T-Caps can help our tracker to get a more accurate appearance even the target object undergoes heavy occlusion. Besides, by adding the proposed T-Caps (see Figure 12b), the response maps change smoothly when the appearance of the target changes drastically. These prove that the aggregation of the temporal relationships facilitates to generate more accurate response.

PGR. To demonstrate the effectiveness of PGR, we explore the difference between "model $w/$ EMR" and " model $w/$ PGR", where "EMR" and "PGR" indicates the original EM routing and prior-guided capsule routing algorithm, respectively. The performance comparison of these two variations are shown in Table 4(2,3), which indicates that with the guidance of the initial capsules generated from the first frame, thereby grabbing more satisfactory spatial relationships. Especially, we are pleased to find that PGR is more effective for long-term tracking. Figure 13 exhibits some complex scenes to show the superiority of PGR.

KDPMC. To show the superiority of KDPMC, we compare



(a) (b)

Figure 14: (a) Visual comparisons for the KDPMC on the $David3$ sequence. (b) Plot of tracking accuracy vs. speed.

the framework with a modified version that is obtained by adding KDPMC strategy. Table 4(4) and Figure 14 show the quantitative and visual comparisons, respectively. It is obvious from Table 4(4) that our KDPMC achieves almost similar performance compared to the one w/o KDPMC both in OTB100 and LaSOT datasets. Additionally, in Figure 14a, we can find that KDPMC helps to preserve more discriminative information as well as reduce noise. Besides, as observed in Figure 14b, without considering the initialization, the processes of both 8-D PM w/o and $w/$ KDPMC are same, their speeds are same, i.e., 27 FPS. That is, our KDPMC greatly elevates the speed to 27 FPS with a slight accuracy drop. By considering the initialization (i.e., 8-D PM w/o KDPMC fine-tuned on the 1^{st} frame, 8-D PM $w/$ KDPMC fine-tuned on the 1^{st} frame and compressed through the KDPMC from 16-D PM), their speeds are 24.1 and 21.6 FPS, respectively.

6. Conclusions

In this paper, we propose a relationships-aware regression tracking for the problem of tracking drift caused by missing parts of the object, which is resorting by the relationships from CapsNet. To achieve this, we exploit spatial-temporal relationships from the proposed S-Caps, T-Caps, and STCaps Learning. Besides, a prior-guided routing algorithm is designed for guiding capsule assignments of subsequent frames more accurately. To reduce the heavy computation burden of the aforementioned modules for meeting the purpose of fast-tracking, where a knowledge distillation pose matrix compression strategy is adopted for these modules. Experimental results demonstrate that our method achieves competitive results while operating at over 27 FPS.

Acknowledgements: This work was supported in part by the National Key R&D Program of China under Grant 2018YFC0832304 and 2020AAA0106502, by the Natural Science Foundation of China under Grant 62073105, by the Distinguished Youth Science Foundation of Heilongjiang Province of China under Grant JC2018021, by the State Key Laboratory of Robotics and System (HIT) under Grant SKLRS-2019-KF-14 and SKLRS-202003D, and by the Heilongjiang Touyan Innovation Team Program.

References

[1] Haoli Bai, Jiaxiang Wu, Irwin King, and Michael Lyu. Few shot network compression via cross distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3203–3210, 2020.

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6182–6191, 2019.

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. In *European Conference on Computer Vision*, pages 205–221. Springer, 2020.

[4] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010.

[5] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6668–6677, 2020.

[6] Zhizhen Chi, Hongyang Li, Huchuan Lu, and Ming-Hsuan Yang. Dual deep network for visual tracking. *IEEE Transactions on Image Processing*, 26(4):2005–2015, 2017.

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4660–4669, 2019.

[8] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.

[9] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7183–7192, 2020.

[10] Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. Capsulevos: Semi-supervised video object segmentation using capsule routing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8480–8489, 2019.

[11] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5374–5383, 2019.

[12] Andrew Gardner, Jinko Kanno, Christian A Duncan, and Rastko Selmic. Measuring distance between unordered sets of different sizes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 137–143, 2014.

[13] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 1763–1771, 2017.

[14] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

[15] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[16] Ziyuan Huang, Changhong Fu, Yiming Li, Fuling Lin, and Peng Lu. Learning aberrance repressed correlation filters for real-time uav tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2891–2900, 2019.

[17] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1125–1134, 2017.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Cehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, et al. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[20] Rodney LaLonde and Ulas Bagci. Capsules for object segmentation. *arXiv preprint arXiv:1804.04241*, 2018.

[21] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[22] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Gradnet: Gradient-guided network for visual object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6162–6171, 2019.

[23] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14639–14647, 2020.

[24] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: Algorithms and benchmark. *IEEE Transactions on Image Processing*, 24(12):5630–5644, 2015.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[26] Yi Liu, Qiang Zhang, Dingwen Zhang, and Jungong Han. Employing deep part-object relationships for salient object

detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1232–1241, 2019.

[27] Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep regression tracking with shrinkage loss. In *Proceedings of the European conference on computer vision (ECCV)*, pages 353–369, 2018.

[28] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[29] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

[30] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016.

[31] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014.

[33] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 2555–2564, 2017.

[34] Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. *arXiv preprint arXiv:1506.07452*, 2015.

[35] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2805–2813, 2017.

[36] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6578–6588, 2020.

[37] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6288–6297, 2020.

[38] Guangting Wang, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. Spm-tracker: Series-parallel matching for real-time visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3643–3652, 2019.

[39] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3119–3127, 2015.

[40] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1328–1338, 2019.

[41] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(9):1834–1848, 2015.

[42] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Joint group feature selection and discriminative filter learning for robust visual object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7950–7960, 2019.

[43] Tianyu Yang, Pengfei Xu, Runbo Hu, Hua Chai, and Antoni B Chan. Roam: Recurrently optimizing tracking model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6718–6727, 2020.

[44] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R Scott. Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6728–6737, 2020.

[45] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4591–4600, 2019.

[46] Linyu Zheng, Ming Tang, Yingying Chen, Jinqiao Wang, and Hanqing Lu. Learning feature embeddings for discriminant model based tracking. *arXiv e-prints*, pages arXiv–1906, 2019.