Multi-Stream Deep Similarity Learning Networks for Visual Tracking

Kunpeng Li[1], **Yu Kong**[1] and **Yun Fu**[1,2]
[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA
[2]College of Computer and Information Science, Northeastern University, Boston, MA 02115, USA
{kunpengli,yukong,yunfu}@ece.neu.edu

Abstract

Visual tracking has achieved remarkable success in recent decades, but it remains a challenging problem due to appearance variations over time and complex cluttered background. In this paper, we adopt a tracking-by-verification scheme to overcome these challenges by determining the patch in the subsequent frame that is most similar to the target template and distinctive to the background context. A multi-stream deep similarity learning network is proposed to learn the similarity comparison model. The loss function of our network encourages the distance between a positive patch in the search region and the target template to be smaller than that between positive patch and the background patches. Within the learned feature space, even if the distance between positive patches becomes large caused by the appearance change or interference of background clutter, our method can use the relative distance to distinguish the target robustly. Besides, the learned model is directly used for tracking with no need of model updating, parameter fine-tuning and can run at 45 fps on a single GPU. Our tracker achieves state-of-the-art performance on the visual tracking benchmark compared with other recent real-time-speed trackers, and shows better capability in handling background clutter, occlusion and appearance change.

1 Introduction

Online single-object tracking plays a crucial role in many artificial intelligence applications, such as autonomous driving systems, robotics, human-computer interaction, etc. Given an arbitrary target object marked with a bounding box at the beginning of a video, the goal of single-object visual tracking is to localize this target in subsequent video frames. Due to the requirements of these practical AI systems, an ideal tracker should be fast, accurate and robust to the appearance change of the target caused by illumination variations, occlusion, deformation. However, despite great progress in recent decades, visual tracking remains a challenging problem.

To deal with the appearance change of the target, the most popular trackers learn robust representations of the target object during an online tracking procedure and use them to match the target across frames. This strategy plays a crucial role in visual tracking algorithms such as IVT [Ross *et al.*, 2008], TLD [Kalal *et al.*, 2012]. However, these trackers are trained entirely online. Only simple models can be learned in this way without taking advantage of a large number of videos that are available offline. These offline videos include abundant various scenarios that can help teach the tracker to handle complex challenges in the real world.

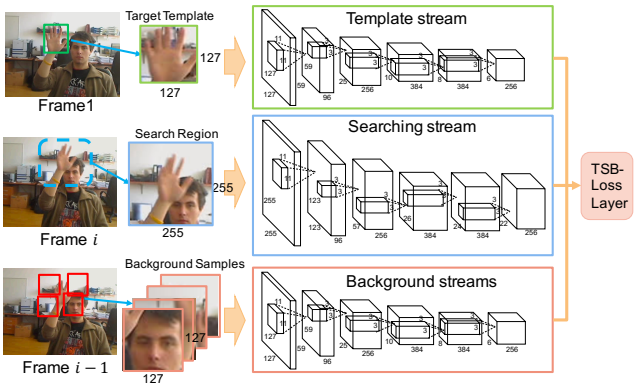

Figure 1: Our multi-stream similarity learning network includes several streams sharing parameters with each other. The definition and input of each stream are annotated in the figure. Details of our Template-searching-background (TSB) loss are shown in Figure 2. It encourages the distance between a positive patch in the search region and the target template to be smaller than the distance between the positive patch and background patches.

Recently, Convolutional Neural Networks (CNN) have been adopted to learn complex models and robust representations from large-scale datasets [Russakovsky *et al.*, 2015]. It achieves superior progress on various artificial intelligence tasks, e.g. object classification [Krizhevsky *et al.*, 2012] and natural language processing. Motivated by the great success of CNNs, several recent works [Nam and Han, 2016] and [Fan and Ling, 2016] treat the tracking problem as a binary classification problem to separate the object from the background. They use pretrained CNN models from classification tasks to exploit the representation power of CNN. During tracking procedure, stochastic gradient descent is applied to fine-tune the network to classify the object and the background for a specific video. However, these methods are too

slow to operate in real-time.

In this work, to learn a robust model entirely offline using CNN for real-time tracking, we treat the single object tracking as a verification problem rather than a classification problem. We learn a general similarity comparison model to verify which image patch in the subsequent frame includes the target object that is marked by the bounding box in the initial frame of the video. Besides, features generated by a general object classifier such as CNN pretrained on ImageNet dataset are insufficient to capture large appearance variations. The underlying reason is the classifier can only tell the difference between different classes and not be able to distinguish different objects of the same type. In order to track a particular target object, similarity learning for comparison at the instance level is performed in our method.

To this end, we propose a multi-stream deep neural network to learn this general similarity comparison model used for tracking by verification. During offline training procedure, instead of learning a detector or classifier for each object in training videos, our model focuses on the appearance variations of the target objects. We optimize our comparison model by introducing a loss function to encourage the distance between a positive patch in the search region and the target template to be smaller than the distance between positive patch and the background patches. In this way, even when the object suffers from huge appearance change or interference of cluttered background, we can still use this relative distance to verify the object from the background context robustly. Once the comparison model has been learned, it can be directly used for tracking. Given the object being tracked in the first frame as a target template, we can go through all possible locations in the search region, and use our similarity comparison model to find a candidate in the subsequent frames that is most similar to the template and discriminative to the background patches. In this way, model updating, parameter fine tuning, forget mechanisms are not required for our tracker.

In summary, we make the following contributions:

- We adopt a tracking-by-verification scheme to verify which image patch in the subsequent frame is most similar to the template and distinctive to the background. Instead of transferring feature space from a general object classifier, we aim to efficiently learn a feature space from a large-scale offline dataset for similarity comparison at the instance level other than the class level.

- We propose a multi-stream deep similarity learning network for verification. By minimizing a relative distance considering both the template and background patches, the loss function of our network could learn a feature space to make the tracker more robust to appearance change and background clutter.

- Extensive experiments demonstrate that our tracker (MDSLT) achieves state-of-the-art performance in the visual tracking benchmark compared with recent real-time-speed trackers. Especially it can better handle cluttered background and huge appearance change.

2 Related Work

Recently, generative trackers have been proposed to match the target in each frame. These trackers aim to develop image representations which can describe the target object robust to appearance changes. Generative trackers assume that the object being tracked can be described by some generative process and hence tracking corresponds to finding the most probable candidate among all possible positions. In [Ross *et al.*, 2008], Eigen Images of the target are computed by incremental PCA over the target intensity-value template. TL1T [Mei and Ling, 2009] assumes that the tracked object can be represented by a sparse combination of over-complete basis vectors. However, these generative trackers could only work well under less complex environments.

To handle challenges of visual tracking and achieve robust performance, composite trackers have been proposed [Li and Zhang, 2014]. TLD [Kalal *et al.*, 2012] aims at using labeled and unlabeled examples for discriminative classifier learning. It applies tracking by combining the results of a detector and an optical flow tracker. For MEEM [Zhang *et al.*, 2014], a discriminative tracker is learned and updated during the tracking process. A set of historical snapshots constitute an expert ensemble to help solve the model drift problem. More recently, MUSTer [Hong *et al.*, 2015] takes advantage of both short and long-term memory to process target appearance memories. They are based on integrated correlation filters and RANSAC matching respectively. However, these trackers are traditionally trained entirely online. Only simple models can be learned by this way without taking advantage of a large number of videos that are available offline.

Motivated by the great success of deep neural network models in related artificial intelligence area, some researchers have tried to adopt deep learning methods for object tracking. On the one hand, several Recurrent Neural Networks based trackers [Gan *et al.*, 2015] have been proposed to take advantage of RNN'S ability of processing data sequences. These methods have not achieve competitive performance on recent popular benchmarks. On the other hand, by taking advantage of large-scale offline training dataset for other similar computer vision problems such as object classification and detection, MDNet [Nam and Han, 2016] trains a CNN as a object detector in an offline phase, then use stochastic gradient descent to learn a detector with patch examples extracted from the video at test time. However, due to the heavy computational burden of online training, these methods are too slow to operate in real-time. Recent deep neural networks based trackers with high performance could only run at around 1 fps on a GPU. Since our deep learning based tracker is trained offline and does not need model updating and online training, it can perform 45 fps on a single GPU.

Similarity comparison model for image patches [Chechik *et al.*, 2009] draws a lot of intention nowadays. Deep neural network based model such as Siamese network [Bromley *et al.*, 1993] is first proposed for signature verification. Recently, these two-stream Siamese architecture have been widely studied and adopted to solve face verification problems [Taigman *et al.*, 2014] and image patch similarity learning [Zagoruyko and Komodakis, 2015]. Motivated by these

achievements, some works propose similar two-stream neural network based trackers to learn the relationship between input pairs. [Held *et al.*, 2016] learns a regression function to locate the objects in current frame by comparing with the previous frame. The strong tie between adjacent frames demands that the tracked object must represent at each frame. Besides, once the tracker fails to locate the object, it will keep failing until the object come back to the same location. [Tao *et al.*, 2016] and [Bertinetto *et al.*, 2016] alternatively take the object marked in the first frame as a template and search the most similar image patch in the current frame to locate the tracked object. However, only comparing with the template in the first frame leads to not robust enough to huge appearance change or background clutter. Instead, our method can learn a similarity comparison model by minimizing a relative distance considering template and background patches. The learned feature space can make our tracker more robust.

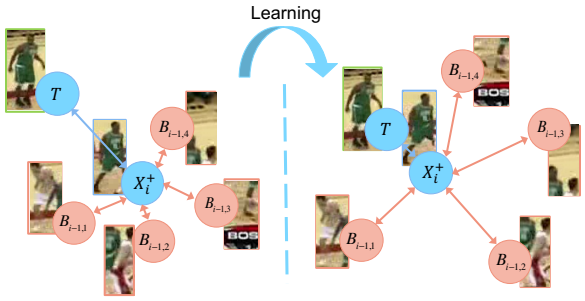

Figure 2: Embedding vectors f of deep neural networks are trained to satisfy the constraints of our template-searching-background loss. By minimizing a relative distance, the loss pulls positive example in the search region closer to the template, meanwhile, pushes $M-1$ background samples away from the positive sample. In this way, even when the object suffers from huge appearance change or interference of cluttered background, we can still use the relative distance to verify the object from the background context robustly.

3 Methods

Arbitrary single-object visual tracking problem is formulated as a verification problem in this work. Given the target object to be tracked in the first frame, we can go through all possible locations in the search region of the subsequent frame to find a candidate that is most similar to the template and discriminative to the background.

3.1 Multi-Stream Similarity Learning Networks

Our goal is to learn a feature space that the object patch of the current frame is closer to the target template when comparing with patches of the background context surrounding the object. We propose a multi-stream similarity learning network to learn such a feature space. As shown in Figure 1, our multi-stream tracking network includes several streams sharing parameters with each other. To avoid confusion, input of Template stream is the target template patch marked in the first frame. Searching stream takes the search region patch of the subsequent frame i as input. We define the remaining streams as background streams. Their inputs are background samples around the object in the frame $i-1$. We only show one background stream here.

In order to track in a fast speed, we design the base networks of our multi-stream tracking framework to be small. They are similar to AlexNet [Krizhevsky *et al.*, 2012] at convolution stage. However, we use less max pooling layers. Because max pooling operation only keeps the strongest activations from neighboring groups of neurons in the same kernel map to use as input for the subsequent layers. In this way, it reduces redundant information and works pretty well in object classification problem, where objects in the same category have big differences in appearance. But in our case, during the tracking procedure, we only track one particular object and sometimes have to distinguish different objects of the same type. Therefore, similarity comparison at the instance level is needed. Since the layers in a deep network capture progressively more abstract representations [Zeiler and Fergus, 2014], it is important to keep sufficient information of high level visual features for tracking by verification. Hence, we add max pooling layers only after the first two convolutional layers, which get activated the lower level visual patterns. Besides, ReLu and batch normalization [Ioffe and Szegedy, 2015] layers are added behindhand except for the final convolutional layer.

3.2 Template-Searching-Background Loss

We define the CNN as an embedding function: given an image patch X, the output of the last layer could be defined as embedding vector $f(X)$. Then we describe the distance of two image patches X_1 and X_2 using cosine distance in the feature space as:

$$D(X_1, X_2) = 1 - \frac{f(X_1) \cdot f(X_2)}{\|f(X_1)\| \, \|f(X_2)\|}. \tag{1}$$

We aim to train our multi-stream tracking framework to obtain a feature representation $f(\cdot)$, so that in the feature space, the distance between target template T in the first frame and positive candidate X_i^+ in the searching region of frame i is small. Meanwhile, the distance between positive candidate X_i^+ and background sample $B_{i,k}$ near the object in frame $i-1$ is encouraged to be larger. Besides, for negative candidate X_i^- in the searching region of frame i, we encourage an opposite trend. Therefore, for each training tracking video, we aim to enforce the following relationship:

$$\begin{aligned} &\begin{cases} D(T, X_i^+) < D(X_i^+, B_{i-1,j}), \\ D(T, X_i^-) > D(X_i^-, B_{i-1,j}), \end{cases} \\ &\forall (X_i^+, X_i^-) \in S_i, \forall B_{i-1,j} \in \beta_{i-1}, \end{aligned} \tag{2}$$

where S_i is the search region of frame i, and β_{i-1} is the set of background samples around the object in frame $i-1$.

Using the logistic loss, we design the loss function as follows:

$$\begin{aligned} &L(T, X_i, \{B_{i-1,j}\}_{j=1}^M) = \log(1 + e^{l_i d_i}) \\ &\text{with } \; d_i = \sum_{j=1}^{M} D(T, X_i) - D(X_i, B_{i-1,j}), \end{aligned} \tag{3}$$

where X_i is one of the candidate patches in the searching region of frame i, $l_i \in \{+1, -1\}$ is the ground-truth label

of X_i, which represents the positive and negative candidate. d_i is the distance relationship function. M is the number of background samples $B_{i-1,j}$ near the object in the frame $i-1$.

In our tracking network, we need to evaluate a lot of candidate patches in the search region of the current frame. A straightforward way is to pass each candidate patch through the network independently. However, this would cause severe computation cost, especially when there are significant overlaps between the candidate patches. Since our network is fully-convolutional, we can set the whole search region S_i as the input of the network and adopt a sliding window on the feature map $f(S_i)$. Then, for each sub-window, we obtain the feature map $f(X_{i,k})$ of the corresponding candidate patch $X_{i,k}$ in the search region S_i. In this way, we only need to pass through the network once. After we get the feature map of each candidate patch, we define our final loss function as follows:

$$L(T, \{X_{i,k}\}_{k=1}^{N}, \{B_{i-1,j}\}_{j=1}^{M}) = \sum_{k=1}^{N} L(T, X_{i,k}, \{B_{i-1,j}\}_{j=1}^{M}), \tag{4}$$

where N is the number of candidate patches $X_{i,k}$ in the search region of frame i.

3.3 Tracking Method

After training our multi-stream similarity learning network using the template-searching-background loss, we can directly adopt it for tracking. For each testing video, the object patch in the given bounding box is set as the target template. During tracking procedure, for any frame i, we set a search region centered on the location of the object which is detected in the $i-1$ frame. To handle scale change of the object, we use different scales searching windows over three scales $1.038^{-1,0,1}$ of its previous size to get crops. Besides, we compute background context patches by sampling around the predicted object location we obtain in the previous frame $i-1$. These patches are passed through the corresponding streams of our network. The candidate is picked if it matches best to the target template and it is discriminative to the background samples in the search region.

$$p_i = \arg\min_{X_{i,k} \in S_i} d(T, X_{i,k}, \{B_{i-1,k}\}_{k=1}^{M}), \tag{5}$$

where $X_{i,k}$ represents one of the candidate patches in the search region S_i of frame i. $d(\cdot)$ is the distance relationship function same as defined in Equation 3.

4 Experiments

4.1 Experimental Setup

Test dataset and evaluation metrics. To evaluate the tracking performance, we test and analyze our MDSLT method on a large benchmark dataset (OTB-100) [Wu *et al.*, 2015]. OTB-100 includes 100 videos with bounding box annotations on each frame. This dataset is large enough to cover various challenging aspects of object tracking, such as low resolution, rotation, background clutter and occlusion. Besides, these aspects are labeled for each video sequences, which help us to analyze the performance details. For evaluation metrics, we follow the evaluation protocol of [Wu *et al.*, 2013], where two metrics are used: success plot and precision plot. Both metrics measure the percentage of successfully tracked frames. For the success plot, a frame is declared to be successfully tracked if the estimated bounding box and the ground truth box have an intersection-over-union overlap larger than a certain threshold. For precision plot, tracking on a frame is considered successful if the distance between the centers of the predicted box and the ground truth box is under a threshold. A plot is given by varying the threshold values. Tracking algorithms are ranked based on the area under curve (AUC) score for the success plot and precision at threshold 20 pixels for the precision plot. We use the available toolkit provided by the benchmark to generate plots and numbers.

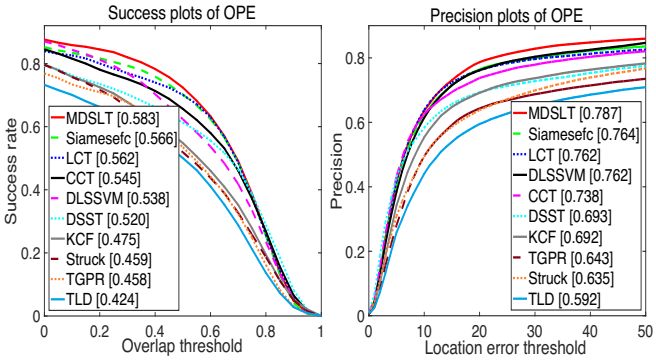

Figure 3: Overlap success plots and distance precision plots over 100 benchmark sequences of OTB-100 using one-pass evaluation (OPE) for the top 10 trackers. The legend contains the area-under-the-curve (AUC) score. Our MDLST method achieves the best performance compared with recent real-time-speed trackers.

Implementation details. For network training, large scale dataset with ground truth bounding box on the target is needed. Commonly used dataset in tracking community such as ALOV [Smeulders and Shah, 2014], VOT [Kristan *et al.*, 2016] are small and have overlap with our test tracking benchmark OTB-100 [Wu *et al.*, 2015], therefore not ideal enough for training. ImageNet Video dataset from [Russakovsky *et al.*, 2015] has been introduced to the tracking community recently. It includes around 4400 videos with more than one million frames annotated by bounding boxes on the target object. Besides, the objects and scenes in this dataset are different from those in the commonly used tracking benchmarks, which can avoid over-fitting. Therefore, we use 90% videos in this dataset for training and the rest 10% for validation. In this way, our training dataset has no overlap with the test dataset. We train our network for 240K iterations with learning rates 0.01 at the beginning and reduce the learning rate by a factor of 10 at every 80K iterations. We use mini-batches of size 8. During the training process, for each video, we set the object patch from a random frame as the target template. Then we get the search region and background samples from another two random adjacent frames $i-1$ and i. For the labels of samples in the search region of frame i, a candidate is considered as a positive sample if it has overlap larger than 0.7 with the corresponding ground truth bounding box, otherwise

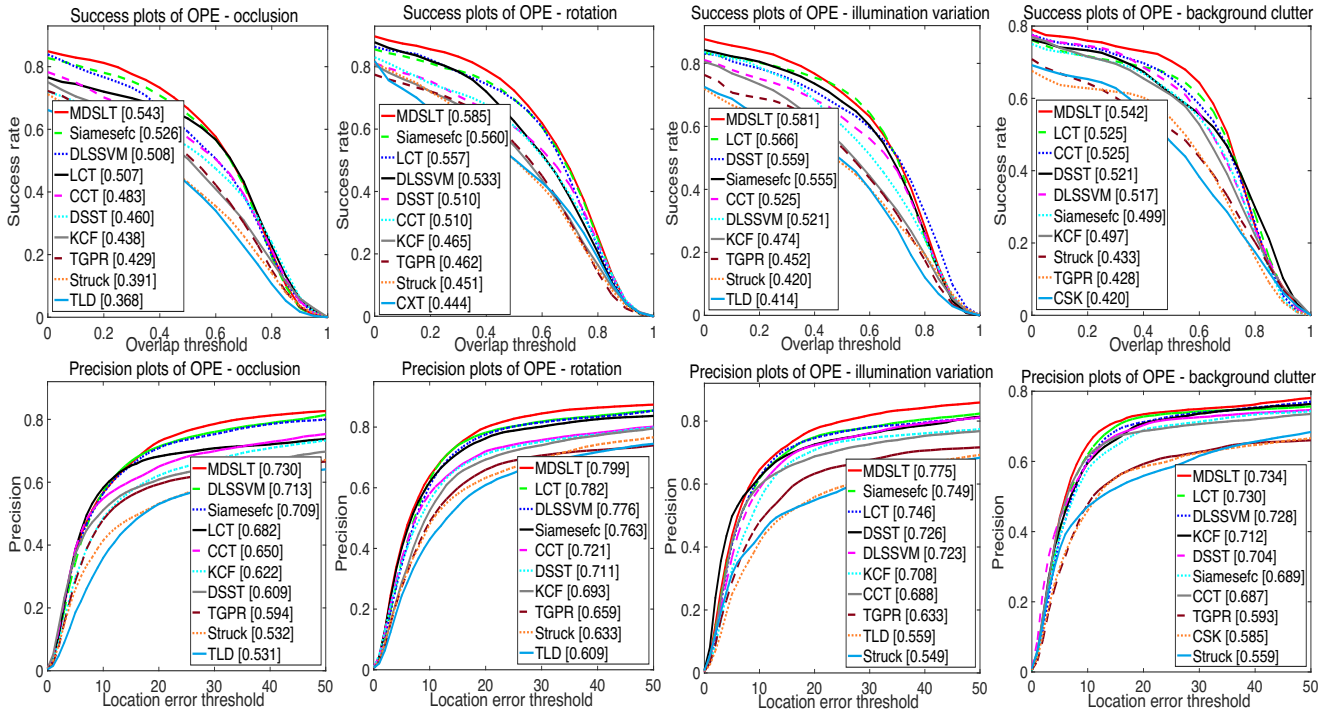

Figure 4: Overlap success plots and distance precision plots over four tracking challenges of occlusion, rotation, illumination variation and background clutter for the top 10 trackers. The legend contains the area-under-the-curve (AUC) score for each tracker. Our MDLST method achieves the best performance when evaluating with four challenging factors.

as negative sample. For background context patches, we use four angular divisions and get four samples around the object in the frame $i-1$ with 0.3 to 0.5 overlap with the ground truth bounding box.

4.2 Results and Analysis

Overall quantitative evaluation. We evaluate our tracker on the benchmark compared with some state-of-the-art tracking methods. In addition to the 31 trackers included in the benchmark [Wu *et al.*, 2015], e.g., TLD [Kalal *et al.*, 2012] and Struck [Hare *et al.*, 2016], we also include some most recent popular trackers presented in the major computer vision and AI conferences, that can run at real-time speed (fps>20) for fair comparison such as SiameseFC [Bertinetto *et al.*, 2016], DSST [Danelljan *et al.*, 2016], DLSSVM [Ning and Yang, 2016], KCF [Henriques *et al.*, 2015], CCT [Zhu *et al.*, 2015] and LCT [Ma *et al.*, 2015]. We report the overall performance results in one-pass evaluation using the distance precision and overlap success rate in Figure 3. For clarity, only the top performing trackers are shown. We can find that our proposed method achieves the best performance in both distance precision (DP) plot and overlap success (OS) plot. For fair comparison, we do not plot the results of MDNet [Nam and Han, 2016], SANet [Fan and Ling, 2016] and HDT [Qi *et al.*, 2016] here, which can only run at around 1 fps. Though they could achieve better accuracy with AUC score of 67.8%, 69.2% and 65.4% for OS plot reported by the authors, we are focusing on the real-time tracker for practical use which inevitably scarifies the accuracy to redeem the efficiency.

Attribute-based evaluation. Videos in the benchmark dataset [Wu *et al.*, 2015] are annotated with 11 attributes to describe the different challenges in the visual tracking problem. We report results for four main challenging attributes in distance precision (DP) plot and overlap success (OS) plot in Figure 4. Among existing methods, LCT outperforms well with area-under-the-curve (AUC) score 56.6% of overlap success (OS) in illumination variation, 73.0% of distance precision (DP) 52.5% of OS in background clutter while our MDSLT method achieve 58.1%, 73.4% and 54.2% respectively with more faster speed (45 fps) than LCT (27 fps). DLSSVM performs well with DP of 71.3% in occlusion and 77.6% in rotation while SiameseFC achieves OS of 52.6% and 56.0% in these two challenges. Our MDSLT achieves better performance with DP of 73.0%, 79.9% and OS of 54.3%, 58.5% respectively. Besides, compared with SiameseFC, our method's better performance proves the efficiency of using the relative distance defined in Equation 3. Even when the object suffers from huge appearance change due to occlusion, rotation, illumination, we can use the relative distance calculated in our learned feature space to verify the object from the background robustly. Especially, considering the background also makes our tracker more robust to the cluttered background with 5% higher than SiameseFC for both OS and DP scores. Quantitative results for all challenge cases are shown in Table 1. Our proposed MDSLT method outperforms all other methods in 10 out of 11 challenges. It also has competitive performance with only 0.1% lower than DLSSVM in the rest challenge.

Qualitative evaluation. We compare our algorithm with

Table 1: Area-under-the-curve (AUC) Score of Success Plot for Each Challenge Case

Attribute \ Method	TLD	DSST	KCF	Struck	CCT	SiamFC	DLSVM	LCT	MDSLT
Fast Motion	0.430	0.465	0.460	0.467	0.557	0.571	0.542	0.534	**0.573**
Motion Blur	0.430	0.473	0.459	0.463	0.530	0.558	**0.571**	0.533	0.570
In-plane Rotation	0.437	0.510	0.465	0.451	0.510	0.560	0.533	0.557	**0.585**
Out-of-plane Rotation	0.385	0.481	0.450	0.424	0.516	0.540	0.531	0.538	**0.561**
Low Resolution	0.346	0.391	0.305	0.312	0.432	0.655	0.377	0.399	**0.665**
Out of View	0.353	0.385	0.493	0.374	0.440	0.483	0.468	0.452	**0.503**
Deformation	0.352	0.434	0.436	0.383	0.508	0.502	0.504	0.499	**0.514**
Scale Variation	0.387	0.479	0.394	0.402	0.486	0.555	0.465	0.488	**0.569**

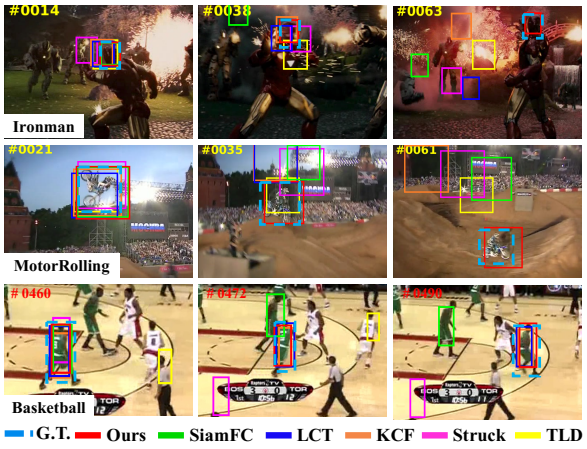

Figure 5: Qualitative result examples of six trackers on three challenging sequences (best view in color). The bounding box with a blue dotted line (marked with G.T.) shows the ground truth for each frame. Our method can track the object more robustly.

other five state-of-the-art trackers, SiameseFC [Bertinetto *et al.*, 2016], LCT [Ma *et al.*, 2015], KCF [Henriques *et al.*, 2015], Struck [Hare *et al.*, 2016], and TLD [Kalal *et al.*, 2012] on several sequences. We show three challenge examples in Figure 5. For *Ironman* sequence, it is hard to track the head of the Iron-man due to the fast motion with significant background clutter such as explosions, robots. All other trackers fail except ours. *MotorRolling* is ideally used to test the tracker's robustness to fast motion, rotation, scale change, motion blur and background clutter. Our method can track the motorcycle more successfully comparing with others. The KCF and LCT tracker is based on a correlation filter learned from HOG features which are less effective to discriminate targets from the cluttered background. The Struck tracker does not perform well since it is less effective in handling appearance change caused by multiple factors with one single classifier. In contrast, our method could learn a more effective feature expression by using CNN trained on large scale offline dataset. Though SimameseFc also uses the deep learning framework, it only considers the similarity between the target in the current frame with the template marked in the first frame. Therefore, SiameseFC fails when this appearance similarity goes down. Since during the training procedure, we optimize our comparison model by introducing a loss function to encourages the distance between a positive patch in the search region and the target template to be smaller than that between positive patch and the background patches. In this way, at test stage, even when the object suffers from huge appearance change due to occlusion, rotation or interference of cluttered background, we can use the relative distance calculated in our learned feature space to verify the object from the background context. Taking background context into consideration also helps our model to distinguish the object from similar objects nearby successfully as shown in *Basketball* sequence. However, by only considering the similarity between target in the current frame with the template, SiameseFC fails and turns to track another similar player with a green shirt.

5 Conclusion

In this paper, we treat the single object tracking as a verification problem rather than a classification or detection problem. Instead of transferring feature space from a general object classifier such as CNN models pretrained on ImageNet, we efficiently learn a feature space from a large-scale offline dataset for similarity comparison at instance level. A multi-stream deep neural network is proposed to learn this general similarity comparison model with a loss function minimizing a relative distance. In this way, considering both template and background information could help to verify and track the object more robustly. The learned model can be directly used for tracking with no need of model updating, parameter fine tuning. From experiments results, our tracker achieves state-of-the-art performance on the visual tracking benchmark compared with other recent real-time-speed trackers. It also shows better capability in handling background clutter, occlusion and appearance change.

Acknowledgments

This work is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484 and U.S. Army Research Office Young Investigator Award W911NF-14-1-0218.

References

[Bertinetto *et al.*, 2016] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865, 2016.

[Bromley *et al.*, 1993] Jane Bromley, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993.

[Chechik *et al.*, 2009] Gal Chechik, Uri Shalit, Varun Sharma, and Samy Bengio. An online algorithm for large scale image similarity learning. In *NIPS*, pages 306–314, 2009.

[Danelljan *et al.*, 2016] Martin Danelljan, Gustav Hager, and Michael Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[Fan and Ling, 2016] Heng Fan and Haibin Ling. Sanet: Structure-aware network for visual tracking. *arXiv:1611.06878*, 2016.

[Gan *et al.*, 2015] Quan Gan, Qipeng Guo, Zheng Zhang, and Kyunghyun Cho. First step toward model-free, anonymous object tracking with recurrent neural networks. *arXiv:1511.06425*, 2015.

[Hare *et al.*, 2016] Sam Hare, Golodetz, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016.

[Held *et al.*, 2016] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[Henriques *et al.*, 2015] João F Henriques, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[Hong *et al.*, 2015] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *CVPR*, pages 749–758, 2015.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[Kalal *et al.*, 2012] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[Kristan *et al.*, 2016] Matej Kristan, Gustavo Fernndez, et al. The visual object tracking challenge results. 2016.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[Li and Zhang, 2014] Kunpeng Li and Xin Zhang. A new fingertip detection and tracking algorithm and its application on writing-in-the-air system. In *CISP*, pages 457–462, 2014.

[Ma *et al.*, 2015] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *CVPR*, pages 5388–5396, 2015.

[Mei and Ling, 2009] Xue Mei and Haibin Ling. Robust visual tracking using $l1$ minimization. In *ICCV*, pages 1436–1443, 2009.

[Nam and Han, 2016] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016.

[Ning and Yang, 2016] Jifeng Ning and Ming-Hsuan Yang. Object tracking via dual linear structured svm and explicit feature map. In *CVPR*, pages 4266–4274, 2016.

[Qi *et al.*, 2016] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, and Jongwoo Lim Ming-Hsuan Yang. Hedged deep tracking. In *CVPR*, 2016.

[Ross *et al.*, 2008] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[Smeulders and Shah, 2014] Arnold WM Smeulders and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[Taigman *et al.*, 2014] Yaniv Taigman, Ming Yang, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[Tao *et al.*, 2016] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *CVPR*, pages 1420–1429, 2016.

[Wu *et al.*, 2013] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418, 2013.

[Wu *et al.*, 2015] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[Zagoruyko and Komodakis, 2015] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, pages 4353–4361, 2015.

[Zeiler and Fergus, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

[Zhang *et al.*, 2014] Jianming Zhang, Shugao Ma, and Stan Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *ECCV*, pages 188–203, 2014.

[Zhu *et al.*, 2015] Guibo Zhu, Jinqiao Wang, Yi Wu, and Hanqing Lu. Collaborative correlation tracking. In *BMVC*, pages 184–1, 2015.