Real-Time MDNet

Ilchae Jung[1], Jeany Son[1], Mooyeol Baek[1], and Bohyung Han[2]

[1]Dept. of CSE, POSTECH, Korea
[2]Dept. of ECE & ASRI, Seoul National University, Korea
{chey0313,jeany,mooyeol}@postech.ac.kr bhhan@snu.ac.kr

Abstract. We present a fast and accurate visual tracking algorithm based on the multi-domain convolutional neural network (MDNet). The proposed approach accelerates feature extraction procedure and learns more discriminative models for instance classification; it enhances representation quality of target and background by maintaining a high resolution feature map with a large receptive field per activation. We also introduce a novel loss term to differentiate foreground instances across multiple domains and learn a more discriminative embedding of target objects with similar semantics. The proposed techniques are integrated into the pipeline of a well known CNN-based visual tracking algorithm, MDNet. We accomplish approximately 25 times speed-up with almost identical accuracy compared to MDNet. Our algorithm is evaluated in multiple popular tracking benchmark datasets including OTB2015, UAV123, and TempleColor, and outperforms the state-of-the-art real-time tracking methods consistently even without dataset-specific parameter tuning.

Keywords: visual tracking, multi-domain learning, RoIAlign, instance embedding loss

1 Introduction

Convolutional Neural Networks (CNNs) are very effective in visual tracking [1–10], but, unfortunately, highly accurate tracking algorithms based on CNNs are often too slow for practical systems. There are only a few methods [11–13] that achieve two potentially conflicting goals—accuracy and speed—at the same time.

MDNet [1] is a popular CNN-based tracking algorithm with state-of-the-art accuracy. This algorithm is inspired by an object detection network, R-CNN [14]; it samples candidate regions, which are passed through a CNN pretrained on a large-scale dataset and fine-tuned at the first frame in a test video. Since every candidate is processed independently, MDNet suffers from high computational complexity in terms of time and space. In addition, while its multi-domain learning framework concentrates on saliency of target against background in each domain, it is not optimized to distinguish potential target instances across multiple domains. Consequently, a learned model by MDNet is not effective to discriminatively represent unseen target objects with similar semantics in test sequences.

A straightforward way to avoid redundant observations and accelerate inference is to perform RoIPooling from a feature map [23], but naïve implementations result in poor localization due to coarse quantization of the feature map. To alleviate such harsh quantization for RoI pooling, [15] proposes RoI alignment (RoIAlign) via bilinear interpolation. However, it may also lose useful localization cues within target if the size of RoI is large. On the other hand, since most CNNs are pretrained for image classification tasks, the networks are competitive to predict semantic image labels but insensitive to tell differences between object instances in low- or mid-level representations. A direct application of such CNNs to visual tracking often yields degradation of accuracy since the embedding generated by pretrained CNNs for image classification task is not effective to differentiate two objects in the same category.

To tackle such critical limitations, we propose a novel real-time visual tracking algorithm based on MDNet by making the following contributions. First, we employ an RoIAlign layer to extract object representations from a preceding fully convolutional feature map. To maintain object representation capacity, the network architecture is updated to construct a high resolution feature map and enlarge the receptive field of each activation. The former is helpful to represent candidate objects more precisely, and the latter is to learn rich semantic information of target. Second, we introduce an instance embedding loss in pretraining stage and aggregate to the existing binary foreground/background classification loss employed in the original MDNet. The new loss function plays an important role to embed observed target instances apart from each other in a latent space. It enables us to learn more discriminative representations of unseen objects even in the case that they have identical class labels or similar semantics.

Our main contributions are summarized as follows:

- We propose a real-time tracking algorithm inspired by MDNet and Fast R-CNN, where an improved RoIAlign technique is employed to extract more accurate representations of targets and candidates from a feature map and improve target localization.
- We learn shared representations using a multi-task loss in a similar way to MDNet, but learn an embedding space to discriminate object instances with similar semantics across multiple domains more effectively.
- The proposed algorithm demonstrates outstanding performance in multiple benchmark datasets without dataset-specific parameter tuning. Our tracker runs real-time with 25 times speed-up compared to MDNet while maintaining almost identical accuracy.

The rest of the paper is organized as follows. We first discuss related work in Section 2. Section 3 discusses our main contribution for target representation via the improved RoIAlign and the instance embedding loss. We present overall tracking algorithm in Section 4, and provide experimental results in Section 5. We conclude this paper in Section 6.

2 Related Work

2.1 Visual Tracking Algorithms

CNN-based visual tracking algorithms typically formulate object tracking as discriminative object detection problems. Some methods [1, 2, 5, 6, 9, 10] draw a set of samples corresponding to candidate regions and compute their likelihoods independently using CNNs. Recent techniques based on discriminative correlation filters boost accuracy significantly by incorporating representations from deep neural networks [16, 17, 3, 4, 18]. Although various tracking algorithms based on CNNs are successful in terms of accuracy, they often suffer from high computational cost mainly due to critical time consuming components within the methods including feature computation of multiple samples, backpropagation for model updates, feature extraction from deep networks, etc. While some CNN-based techniques [19, 20, 13] for visual tracking run real-time by employing offline representation learning without online model updates, their accuracy is not competitive compared to the state-of-the-art methods.

There are only a few real-time trackers [3, 11, 12] that present competitive accuracy. Galoogahi *et al.* [11] incorporate background region to learn more discriminative correlation filters using hand-crafted features. Fan *et al.* [12] design a robust tracking algorithm through interactions between a tracker and a verifier. Tracker estimates target states based on the observation using hand-crafted features efficiently while verifier double-checks the estimation using the features from deep neural networks. Danelljan *et al.* [3] propose a discriminative correlation filter for efficient tracking by integrating multi-resolution deep features. Since its implementation with deep representations is computationally expensive, they also introduce a high-speed tracking algorithm with competitive accuracy based on hand-crafted features. Note that most real-time trackers with competitive accuracy rely on hand-crafted features or limited use of deep representations. Contrary to such real-time tracking methods, our algorithm has a simpler inference pipeline within a pure deep neural network framework.

2.2 Representation Learning for Visual Tracking

MDNet [1] pretrains class-agnostic representations appropriate for visual tracking task by fine-tuning a CNN trained originally for image classification. It deals with label conflict issue across videos by employing multi-domain learning, and achieves the state-of-the-art performance in multiple datasets. Since the great success of MDNet [1], there have been several attempts to learn representations for visual tracking [20–22] using deep neural networks. Bertinetto *et al.* [20] learn to maximize correlation scores between the same objects appearing in different frames. Valmadre *et al.* [21] regress response maps between target objects and input images to maximize the score at the ground-truth target location. Similarly, Gundogdu *et al.* [22] train deep features to minimize difference between the response map from a tracker based on correlation filters and the ground-truth map that has a peaky maximum value at target location.

All the efforts discussed above focus on how to make target objects salient against backgrounds. While this strategy is effective to separate target from background, it is still challenging to discrimninate between object instances with similar semantics. Therefore, our algorithm encourages our network to achieve the two objectives jointly by proposing a novel loss function with two terms.

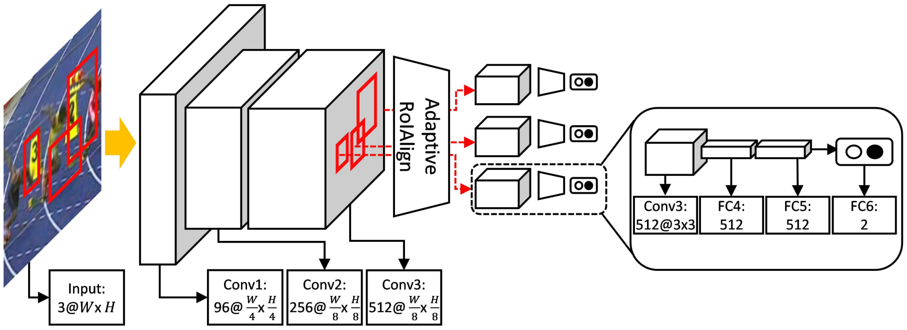

Input:
3@WxH
Conv1:
96@$\frac{W}{4}$x$\frac{H}{4}$
Conv2:
256@$\frac{W}{8}$x$\frac{H}{8}$
Conv3:
512@$\frac{W}{8}$x$\frac{H}{8}$
Adaptive
RoIAlign
Conv3:
512@3x3
FC4:
512
FC5:
512
FC6:
2

Fig. 1. Network architecture of the proposed tracking algorithm. The network is composed of three convolutional layers for extracting a shared feature map, adaptive RoIAlign layer for extracting a specific feature using regions of interest (RoIs), and three fully connected layers for binary classification. The number of channels and the size of each feature map are shown with the name of each layer.

2.3 Feature Extraction in Object Detection

Although R-CNN [14] is successful in object detection, it has significant overhead to extract features from individual regions for inference. Fast R-CNN [23] reduces its computational cost for feature extraction using RoIPooling, which computes fixed-size feature vectors by applying max pooling to the specific regions in a feature map. While the benefit in terms of computational cost is impressive, RoIPooling is not effective to localize targets because it relies on a coarse feature map. To alleviate this limitation, mask R-CNN [15] introduces a new feature extraction technique, RoI alignment (RoIAlign), which approximates features via bilinear interpolation for better object localization. Our work proposes a modified network architecture for an adaptive RoIAlign to extract robust features corresponding to region proposals.

3 Efficient Feature Extraction and Discriminative Feature Learning

This section describes our CNN architecture with an improved RoiAlign layer, which accelerates feature extraction while maintaining quality of representations. We also discuss a novel multi-domain learning approach with discriminative instance embedding of foreground objects.

3.1 Network Architecture

Fig. 1 illustrates architecture of our model. The proposed network consists of fully convolutional layers (`conv1-3`) for constructing a shared feature map, an adaptive RoIAlign layer for extracting feature of each RoI, and three fully connected layers (`fc4-6`) for binary classification. Given a whole image with a set of proposal bounding boxes as an input, the network computes a shared feature map of the input image through a single forward pass. A CNN feature corresponding to each RoI is extracted from the shared feature map using an adaptive RoIAlign operation. Through this feature computation strategy, we reduce computational complexity significantly while improving quality of features.

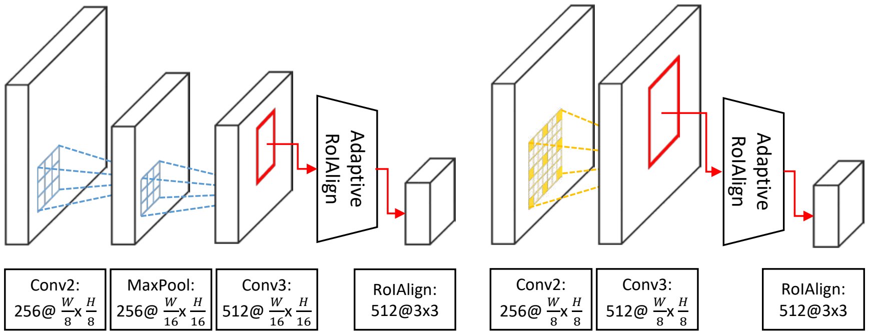
The extracted feature representation from each RoI is fed to two fully connected layers for classification between target and background. We create multiple branches of domain specific layers ($\texttt{fc6}^1$-$\texttt{fc6}^D$) for multi-domain learning, and learn a discriminative instance embedding. During online tracking, a set of the domain-specific fully connected layers are replaced by a single binary classification layer with softmax cross-entropy loss, which will be fine-tuned using the examples from an initial frame.

3.2 Improved RoIAlign for Visual Tracking

Our network has an RoIAlign layer to obtain object representations from a fully convolutional feature map constructed from a whole image. However, features extracted by RoIAlign are inherently coarse compared to the ones from individual proposal bounding boxes. To improve quality of representations of RoIs, we need to construct a feature map with high resolution and rich semantic information. These requirements can be addressed by computing a denser fully convolutional feature map and enlarging the receptive field of each activation. To these ends, we remove a max pooling layer followed by `conv2` layer in VGG-M network [24] and perform dilated convolutions [25] in `conv3` layer with rate $r = 3$. This strategy results in a twice larger feature map than the output of `conv3` layer in the original VGG-M network. It allows to extract high resolution features and improve quality of representation. Fig. 2 compares our network for dense feature map computation with the original VGG-M network.

Our adaptive RoIAlign layer computes more reliable features, especially for large objects, using a modified bilinear interpolation. Since ordinary RoIAlign only utilizes nearby grid points on the feature map to compute the interpolated value, it may lose useful information if the interval of the sampled points for RoI is larger than the one of the feature map grid. To handle this issue, we adjust the interval of the grid points from the shared dense feature map adaptively. In specific, the bandwidth of the bilinear interpolation is determined by the size of RoIs; it is proportional to $[\frac{w}{w'}]$, where w and w' denote the width of RoI after the `conv3` layer and the width of RoI's output feature in the RoIAlign layer, respectively, and $[\cdot]$ is a rounding operator.

The technique integrating a network to employ a dense feature map and the adaptive RoIAlign is referred to as improved RoIAlign. Our adaptive RoIAlign layer produces a 7×7 feature map, and a max pooling layer is applied after the layer to produce a 3×3 feature map. Although the improved RoIAlign makes minor changes, it improves performance of our tracking algorithm significantly in practice. This is partly because, on the contrary to object detection, tracking errors originated from subtle differences in target representations are propagated over time and create large errors to make trackers fail eventually.



(a) Original VGG-M network (b) Network for dense feature map

Fig. 2. Network architecture of the part of our fully convolutional network for extracting a shared feature map. Max pooling layer is removed after conv2 layer in original VGG-M network, and dilated convolution with rate $r = 3$ is applied for extracting a dense feature map with a higher spatial resolution.

3.3 Pretraining for Discriminative Instance Embedding

The goal of our learning algorithm is to train a discriminative feature embedding applicable to multiple domains. MDNet has the separate shared and domain-specific layers to learn the representations distinguishing between target and background. In addition to this objective, we propose a new loss term, referred to as an instance embedding loss, which enforces target objects in different domains to be embedded far from each other in a shared feature space and enables to learn discriminative representations of the unseen target objects in new test sequences. In other words, MDNet only attempts to discriminate target and background in individual domains, and may not be powerful to discriminate foreground objects in different domains, especially when the foreground objects belong to the same semantic class or have similar appearances. This is partly because the original CNNs are trained for image classification. To handle this issue, our algorithm incorporates an additional constraint, which embeds foreground objects from multiple videos to be apart from each other.

Given an input image $\mathbf{x}^d$ in domain d and a bounding box R, the output score of the network, denoted by $\mathbf{f}^d$, is constructed by concatenating the activations from the last fully connected layers ($\texttt{fc6}^1$-$\texttt{fc6}^D$) as

$$\mathbf{f}^d = [\phi^1(\mathbf{x}^d; R), \phi^2(\mathbf{x}^d; R), \ldots, \phi^D(\mathbf{x}^d; R)] \in \mathbb{R}^{2\times D}, \tag{1}$$

where $\phi^d(\cdot;\cdot)$ is a 2D binary classification score from the last fully connected layer $\texttt{fc6}^d$ `in domain` d`,` and D is the number of domains in a training dataset. The output feature is given to a softmax function for binary classification, which determines whether a bounding box R is a target or a background patch in domain d. Additionally, the output feature is passed through another softmax operator for discriminating instances in multiple domains. The two softmax functions are

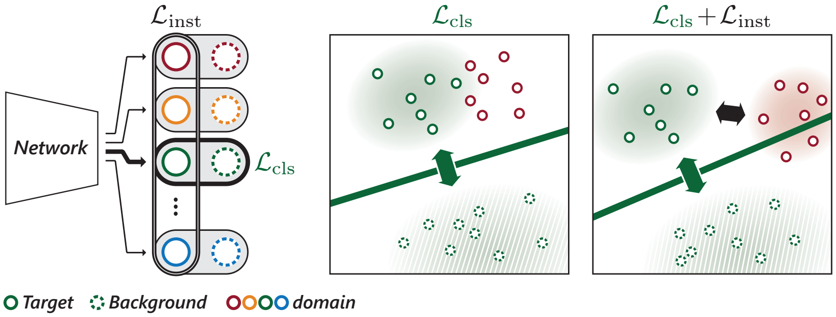

$\mathcal{L}_{\text{inst}}$
Network
$\mathcal{L}_{\text{cls}}$
$\mathcal{L}_{\text{cls}}$
$\mathcal{L}_{\text{cls}}+\mathcal{L}_{\text{inst}}$
Target
Background
domain

Fig. 3. Multi-task learning for binary classification of the target object and instance embedding across multiple domains. The binary classification loss is designed to distinguish the target and background, while the instance embedding loss separates target instances. Note that a minibatch in each iteration for training is constructed by sampling from a single domain.

given by

$$[\sigma_{\text{cls}}(\mathbf{f}^d)]_{ij} = \frac{\exp(f_{ij}^d)}{\sum_{k=1}^{2}\exp(f_{kj}^d)} \quad \text{and} \quad [\sigma_{\text{inst}}(\mathbf{f}^d)]_{ij} = \frac{\exp(f_{ij}^d)}{\sum_{k=1}^{D}\exp(f_{ik}^d)}, \tag{2}$$

where $\sigma_{\text{cls}}(\cdot)$ compares scores of target and background in each domain whereas $\sigma_{\text{inst}}(\cdot)$ compares the positive scores of the objects across all domains.

Our network minimizes a multi-task loss $\mathcal{L}$ on the two softmax operators, which is given by

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \alpha \cdot \mathcal{L}_{\text{inst}}, \tag{3}$$

where $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{inst}}$ are loss terms for binary classification and discriminative instance embedding, respectively, and α is a hyper-parameter that controls balance between the two loss terms. Following MDNet, we handle a single domain in each iteration; the network is updated based on a minibatch collected from the $(k \bmod D)^{\text{th}}$ domain only in the k^{th} iteration.

The binary classification loss with domain $\hat{d}(k) = (k \bmod D)$ in the k^{th} iteration is given by

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{2}[\mathbf{y}_i]_{c\hat{d}(k)} \cdot \log\left(\left[\sigma_{\text{cls}}\left(\mathbf{f}_i^{\hat{d}(k)}\right)\right]_{c\hat{d}(k)}\right), \tag{4}$$

where $\mathbf{y}_i \in \{0,1\}^{2\times D}$ is a one-hot encoding of a ground-truth label; its element $[\mathbf{y}_i]_{cd}$ is 1 if a bounding box R_i in domain d corresponds to class c, otherwise 0. Also, the loss for discriminative instance embedding is given by

$$\mathcal{L}_{\text{inst}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D}[\mathbf{y}_i]_{+d} \cdot \log\left(\left[\sigma_{\text{inst}}\left(\mathbf{f}_i^d\right)\right]_{+d}\right). \tag{5}$$

Note that the instance embedding loss is applied only to positive examples using the positive channel denoted by + in Eq. (5). As a result of the proposed loss, the positive scores of target objects in current domain become larger while their scores in other domains get smaller. It leads to a distinctive feature embedding of target instances and makes it effective to distinguish similar objects potentially appearing in new testing domains.

Fig. 3 illustrates impact of the multi-task learning for discriminative feature embedding of target instances across multiple domains.

4 Online Tracking Algorithm

We discuss the detailed procedure of our tracking algorithm including implementation details. The pipeline of our tracking algorithm is almost identical to MDNet [1].

4.1 Main Loop of Tracking

Once pretraining is completed, we replace multiple branches of domain-specific layers ($\texttt{fc6}^1$-$\texttt{fc6}^D$) with a single branch for each test sequence. Given the first frame with ground-truth of target location, we fine-tune fully connected layers (`fc4-6`) and customize the network to a test sequence. For the rest of frames, we update the fully connected layers in an online manner while convolutional layers are fixed. Given an input frame at time t, a set of samples, denoted by $\{\mathbf{x}_t^i\}_{i=1\dots N}$, are drawn from a Gaussian distribution centered at the target state of the previous frame, the optimal target state is given by

$$\mathbf{x}_t^* = \underset{\mathbf{x}_t^i}{\arg\max}\, f^+(\mathbf{x}_t^i), \tag{6}$$

where $f^+(\mathbf{x}_t^i)$ indicates the positive score of the i^{th} sample drawn from the current frame at time step t. Note that tracking is performed in a three dimensional state space for translation and scale change.

We also train a bounding box regressor to improve target localization accuracy motivated by the success in [1]. Using a set of extracted features from RoIs from the first frame of a video, $\mathcal{F}_i^{\text{RoI}}$, we train a simple linear regressor in the same way to [14, 26]. We apply the learned bounding box regressor from the second frame and adjust the estimated target regions if the estimated target state is sufficiently reliable, $f^+(\mathbf{x}_t^*) > 0.5$.

4.2 Online Model Updates

We perform two complementary update strategies as in MDNet [1]: long-term and short-term updates to maintain robustness and adaptiveness, respectively. Long-term updates are regularly applied using the samples collected for a long period of time, while short-term updates are triggered whenever the score of the estimated target is below a threshold and the result is unreliable.

A minibatch is composed of 128 examples—32 positive and 96 negative samples, for which we employ hard minibatch mining in each iteration of online learning procedure. The hard negative examples are identified by testing 1024 negative examples and selecting the ones with top 96 positive scores.

4.3 Implementation Details

Network initialization and input management The weights of three convolutional layers are transferred from the corresponding parts in VGG-M network [24] pretrained on ImageNet [27] while fully connected layers are initialized randomly. An input image is resized to make the size of target object fit to 107×107, and cropped to the smallest rectangle enclosing all sample RoIs. The receptive field size of a single unit in the last convolutional layer is equal to 75×75.

Offline pretraining For each iteration of offline pretraining, we construct a minibatch with samples collected from a single domain. We first sample 8 frames randomly in the selected domain, and draw 32 positive and 96 negative examples from each frame, which results in 256 positive and 768 negative data altogether in a minibatch. The positive bounding boxes have overlap larger than 0.7 with ground-truths in terms of Intersection over Union (IoU) measure while the negative samples have less than 0.5 IoUs. Instead of backpropagating gradients in each iteration, we accumulate the gradients from backward passes in multiple iterations; the network is updated at every 50 iteration in our experiments. We train our models on ImageNet-Vid [27], which is a large-scale video dataset for object detection. Since this dataset contains a lot of video sequences, almost 4500 videos, we randomly choose 100 videos for an instance embedding loss in each iteration. Hyper-parameter α in Eq. (3) is set to 0.1.

Online training Since pretraining stage aims to learn generic representation for visual tracking, we have to fine-tune the pretrained network at the first frame of each testing video. We draw 500 positive and 5000 negative samples based on the same IoU criteria with the pretraining stage. From the second frame, the training data for online updates are collected after tracking is completed in each frame. The tracker gather 50 positive and 200 negative examples that have larger than 0.7 IoU and less than 0.3 IoU with the estimated target location, respectively. Instead of storing the original image patches, our algorithm keep their feature representations to save time and memory by avoiding redundant computation. Long-term updates are executed every 10 frame.

Optimization Our network is trained by a Stochastic Gradient Descent (SGD) method. For offline representation learning, we train the network for 1000 epochs with learning rate 0.0001 while it is trained for 50 iterations at the first frame of a test video. For online updates, the number of iterations for fine-tuning is

15 and the learning rate is set to 0.0003. The learning rate for `fc6` is 10 times bigger than others (`fc4-5`) to facilitate convergence in practice. The weight decay and momentum are fixed to 0.0005 and 0.9, respectively. Our algorithm is implemented in PyTorch with 3.60 GHz Intel Core I7-6850K and NVIDIA Titan Xp Pascal GPU.

5 Experiments

This section presents our results on multiple benchmark datasets with comparisons to the state-of-the-art tracking algorithms, and analyzes performance of our tracker by ablation studies.

5.1 Evaluation Methodology

We evaluate our tracker, denoted by real-time MDNet or RT-MDNet, on three standard datasets including OTB2015 [28], UAV123 [29] and TempleColor [30]. For comparison, we employ several state-of-the-art trackers including ECO [3], MDNet [1], MDNet+IEL, SRDCF [31], C-COT [4], and top performing real-time trackers, ECO-HC [3], BACF [11], PTAV [12], CFNet [21], SiamFC [20] and DSST [32]. ECO-HC is a real-time variant of ECO based on hand-crafted features, HOG and color names, while MDNet+IEL is a version of MDNet with the instance embedding loss. Both MDNet and MDNet+IEL are pretrained on IMAGENET-VID.

We follow the evaluation protocol presented in a standard benchmark [28], where performance of trackers is evaluated based on two criteria—bounding box overlap ratio and center location error—and is visualized by success and precision plots. The two plots are generated by computing ratios of successfully tracked frames at a set of different thresholds in the two metrics. The Area Under Curve (AUC) scores of individual trackers are used to rank the trackers in the success plot. In the precision plots, the ranks of trackers are determined by the accuracy at 20 pixel threshold. In both plots, real-time trackers are represented with solid lines while the rests are denoted by dashed lines. Note that the parameters of our algorithm are fixed throughout the experiment; we use the same parameters for all three tested datasets while others may have the different parameter setting for each dataset.

5.2 Evaluation on OTB2015

We first analyze our algorithm on OTB2015 dataset [28], which consists of 100 fully annotated videos with various challenging attributes. Fig. 4 presents precision and success plots on OTB2015 dataset.

The results clearly show that real-time MDNet outperforms all the tested real-time trackers significantly in terms of both measures. It also has competitive accuracy compared to the top-ranked trackers while it is approximately 130, 25, and 8 times faster than C-COT, MDNet, and ECO, respectively. Our algorithm

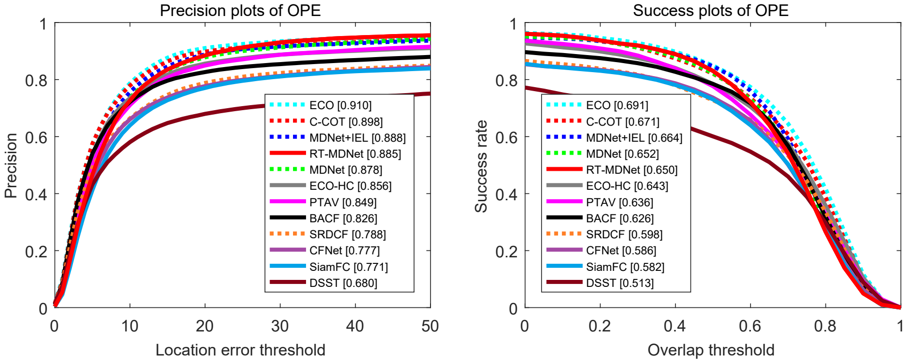


Fig. 4. Quantitative results on OTB2015 [28].

Table 1. Quantitative comparisons of real-time trackers on OTB2015

Trackers	DSST [32]	SiamFC [20]	CFNet [21]	BACF [11]	PTAV [12]	ECO-HC [3]	RT-MDNet
Succ (%)	51.3	58.2	58.6	62.7	63.5	64.3	65.0
Prec (%)	68.0	77.1	77.7	82.7	84.8	85.6	88.5
FPS	24	86	43	35	25	60	46/52

is slightly less accurate than the competitors when the overlap threshold is larger than 0.8. It implies that the estimated target bounding boxes given by our tracker are not very tight compared to other state-of-the-art methods; possible reasons are inherent drawback of CNN-based trackers and the limitation of our RoIAlign for target localization at high precision area.

Table 1 presents overall performance of real-time trackers including our algorithm in terms of AUC for success rate, precision rate at 20 pixel threshold, and speed measured by FPS. The proposed method outperforms all other real-time trackers by substantial margins in terms of two accuracy measures. It runs very fast, 46 FPS in average, while speed except the first frame is approximately 52 FPS. Note that our tracker needs extra computational cost at the first frame for fine-tuning the network and learning a bounding box regressor.

We also illustrate the qualitative results of multiple real-time algorithms on a subset of sequences in Fig. 7. Our approach shows consistently better performance in various challenging scenarios including illumination change, scale variation and background clutter. Some failure cases are presented in Fig. 8. Our algorithm loses target in *Soccer* sequence due to significant occlusion and in *Biker* sequence due to sudden large motion and out-of-plane rotation. Objects with similar appearances make our tracker confused in *Coupon* sequence, and dramatic non-rigid appearance changes in *Jump* cause drift problems.

5.3 Evaluation on TempleColor

Fig. 5 illustrates the precision and success plots on TempleColor dataset [30], which is containing 128 color videos while most of sequences are overlapped with

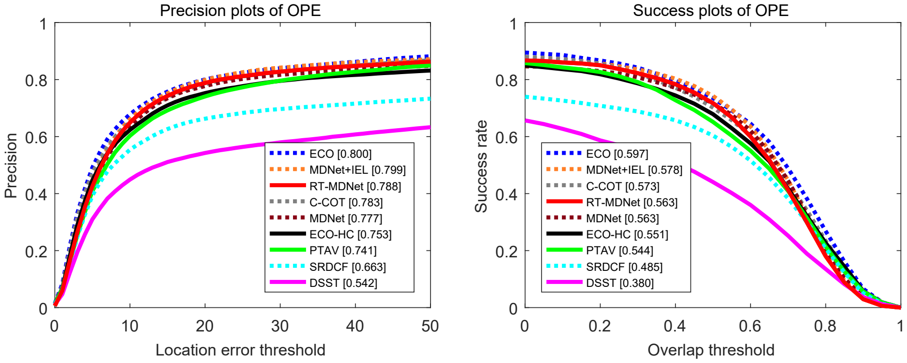


Fig. 5. Quantitative results on TempleColor [30].

Table 2. Impacts of different feature extraction methods on accuracy of RT-MDNet

Pooling Operation	align	adaRoI	denseFM	AUC (%)	Prec (%)
RoIPooling [23]				35.4	53.8
RoIAlign [15]	√			56.1	80.4
Adaptive RoIAlign	√	√		59.0	83.8
RoIAlign with denseFM	√		√	60.7	84.3
Improved RoIAlign	√	√	√	61.9	85.3

OTB2015 dataset [28]. Our method again surpass all real-time trackers[1] and has a substantial improvement over ECO-HC.

5.4 Evaluation on UAV123

We also evaluate real-time MDNet on the aerial video benchmark, UAV123 [29] whose characteristics inherently differ from other datasets such as OTB2015 and TempleColor. It contains 123 aerial videos with more than 110K frames altogether. Fig. 6 illustrates the precision and success plots of the trackers that have publicly available results on this dataset. Surprisingly, in the precision rate, our tracker outperforms all the state-of-the-art methods including non-real-time tracker while it is very competitive in the success rate as well. In particular, our tracker beats ECO, which is a top ranker in OTB2015 and TempleColor, on the both metrics with a approximately 8 times speed-up. It shows that our algorithm has better generalization capability without parameter tuning to a specific dataset.

5.5 Ablation Study

We perform several ablation studies on OTB2015 [28] to investigate the effectiveness of individual components in our tracking algorithm. We first test the

[1] The AUC score of BACF is reported in their paper by 52.0%, which is much lower than the score of our tracker.

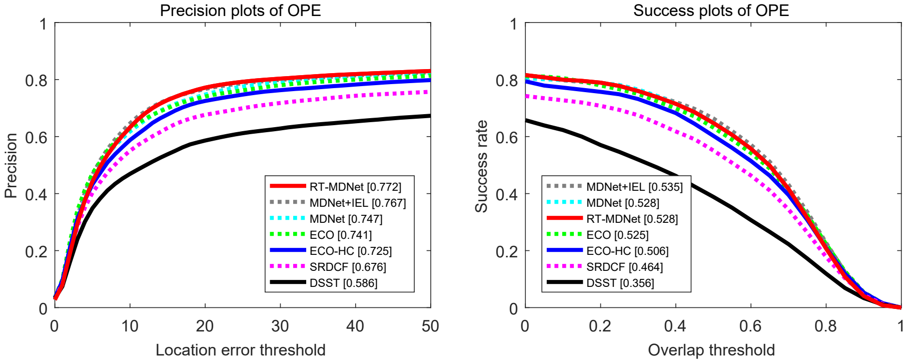


Fig. 6. Quantitative results on UAV123 [29].

Table 3. Internal comparison results pretrained on ImageNet-Vid dataset.

Method	$\mathcal{L}_{\text{cls}}$	$\mathcal{L}_{\text{inst}}$	BBreg	Succ (%)	Prec (%)
Ours–BBR–IEL	√			61.9	84.2
Ours–BBR	√	√		64.1	87.7
Ours	√	√	√	65.0	88.5

impact of the proposed RoIAlign on the quality of our tracking algorithm. For this experiments, we pretrain our network using VOT-OTB dataset, which consist of 58 videos collected from VOT2013 [33], VOT2014 [34] and VOT2015 [35] excluding the videos in OTB2015. Table 2 presents several options to extract target representations, which depend on choice between RoIPooling and RoIAlign, use of adaptive RoIAlign layer (adaRoI) and construction of dense feature map (denseFM). All results consistently support that each component of our improved RoIAlign makes meaningful contribution to tracking performance improvement.

We also investigated two additional versions of our tracking algorithm—one is without bounding box regression (Ours–BBR) and the other is without bounding box regression and instance embedding loss (Ours–BBR–IEL). Table 3 summarizes the results from this internal comparison. According to our experiment, the proposed multi-task loss (binary classification loss and instance embedding loss) and bounding box regression are both helpful to improve localization[2].

6 Conclusions

We presented a novel real-time visual tracking algorithm based on a CNN by learning discriminative representations of target in a multi-domain learning framework. Our algorithm accelerates feature extraction procedure by an improved RoIAlign technique. We employ a multi-task loss effective to discriminate object instances across domains in the learned embedding space. The proposed

[2] As illustrated in Fig. 4, 5, and 6, we verified that applying instance embedding loss to MDNet also improves performances.

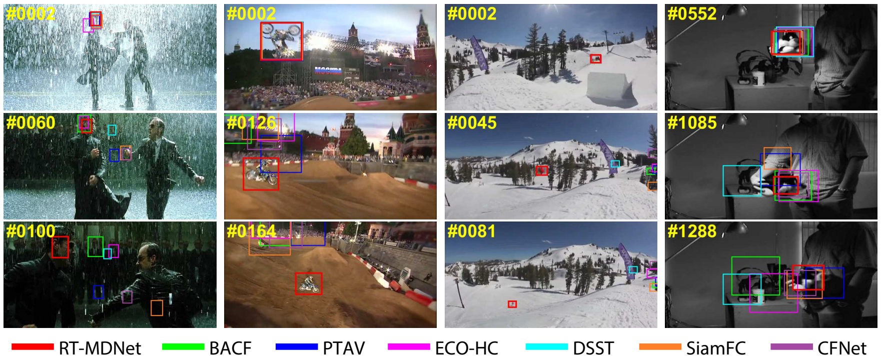


Fig. 7. Qualitative results of the proposed method on several challenging sequences (*Matrix*, *MotorRolling*, *Skiing*, *Sylvester*) in OTB2015 dataset.

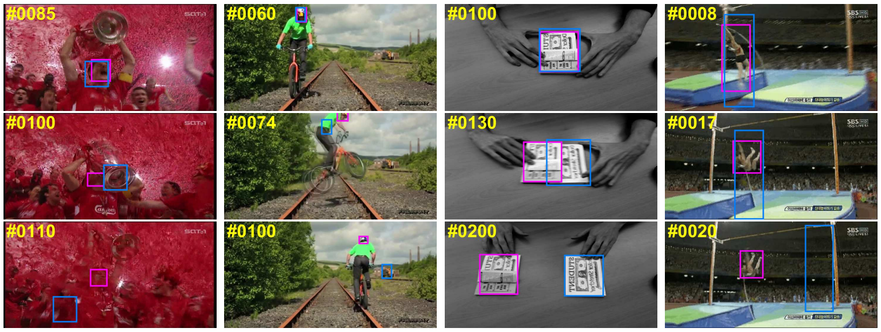


Fig. 8. Failure cases of RT-MDNet in *Soccer, Biker, Coupon,* and *Jump* sequence. Magenta and blue bounding boxes denote ground-truths and our results, respectively.

algorithm was evaluated on the public visual tracking benchmark datasets and demonstrated outstanding performance compared to the state-of-the-art techniques, especially real-time trackers.

Acknowledgement

This research was supported in part by Research Resettlement Fund for the new faculty of Seoul National University and the IITP grant [2014-0-00059, Development of Predictive Visual Intelligence Technology (DeepView); 2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion; 2017-0-01780, The Technology Development for Event Recognition/Relational Reasoning and Learning Knowledge based System for Video Understanding].

References

1. Nam, H., Han, B.: Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. In: CVPR. (2016)
2. Nam, H., Baek, M., Han, B.: Modeling and Propagating CNNs in a Tree Structure for Visual Tracking. arXiv preprint arXiv:1608.07242 (2016)
3. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: ECO: Efficient Convolution Operators for Tracking. In: CVPR. (2017)
4. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In: ECCV. (2016)
5. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J.: Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning. In: CVPR. (2017)
6. Fan, H., Ling, H.: SANet: Structure-Aware Network for Visual Tracking. In: CVPRW. (2017)
7. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual Tracking with Fully Convolutional Networks. In: ICCV. (2015)
8. Hong, S., You, T., Kwak, S., Han, B.: Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network. In: ICML. (2015)
9. Teng, Z., Xing, J., Wang, Q., Lang, C., Feng, S., Jin, Y.: Robust Object Tracking based on Temporal and Spatial Deep Networks. In: ICCV. (2017)
10. Han, B., Sim, J., Adam, H.: Branchout: Regularization for Online Ensemble Tracking with Convolutional Neural Networks. In: CVPR. (2017)
11. Galoogahi, H., Fagg, A., Lucey, S.: Learning Background-Aware Correlation Filters for Visual Tracking. In: ICCV. (2017)
12. Fan, H., Ling, H.: Parallel Tracking and Verifying: A Framework for Real-time and High Accuracy Visual Tracking. In: ICCV. (2017)
13. Huang, C., Lucey, S., Ramanan, D.: Learning Policies for Adaptive Tracking with Deep Feature Cascades. In: ICCV. (2017)
14. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: CVPR. (2014)
15. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV. (2017)
16. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical Convolutional Features for Visual Tracking. In: ICCV. (2015)
17. Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R., Yang, M.H.: CREST: Convolutional Residual Learning for Visual Tracking. In: ICCV. (2017)
18. Zhang, T., Xu, C., Yang, M.H.: Multi-task Correlation Particle Filter for Robust Object Tracking. In: CVPR. (2017)
19. Held, D., Thrun, S., Savarese, S.: Learning to Track at 100 FPS with Deep Regression Networks. In: ECCV. (2016)
20. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.: Fully-Convolutional Siamese Networks for Object Tracking. In: ECCVW. (2016)
21. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.: End-to-end Representation Learning for Correlation Filter based Tracking. In: CVPR. (2017)
22. Gundogdu, E., Alatan, A.A.: Good Features to Correlate for Visual Tracking. TIP (2018)
23. Girshick, R.: Fast R-CNN. In: ICCV. (2015)
24. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the Devil in the Details: Delving Deep into Convolutional Nets. In: BMVC. (2014)
25. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. TPAMI **40**(4) (2017) 834–848

26. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object Detection with Discriminatively Trained Part-based Models. TPAMI **32**(9) (2010) 1627–1645
27. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3) (2015) 211–252
28. Wu, Y., Lim, J., Yang, M.: Object Tracking Benchmark. TPAMI **37**(9) (2015) 1834–1848
29. Mueller, M., Smith, N., Ghanem, B.: A Benchmark and Simulator for UAV Tracking. In: ECCV. (2016)
30. Liang, P., Blasch, E., Ling, H.: Encoding Color Information for Visual Tracking: Algorithms and Benchmark. TIP **24**(12) (2015) 5630–5644
31. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Learning Spatially Regularized Correlation Filters for Visual Tracking. In: ICCV. (2015)
32. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Discriminative Scale Space Tracking. TPAMI **39**(8) (2017) 1561–1575
33. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Porikli, F., Cehovin, L., Nebehay, G., Fernandez, G., Vojir, T., Gatt, A., et al.: The Visual Object Tracking VOT2013 Challenge Results. In: ICCVW. (2013)
34. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Cehovin, L., Nebehay, G., Vojíř, T., Fernandez, G., et al.: The Visual Object Tracking VOT2014 Challenge Results. In: ECCVW. (2014)
35. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernández, G., Vojíř, T., Häger, G., Nebehay, G., Pflugfelder, R., et al.: The Visual Object Tracking VOT2015 Challenge Results. In: ICCVW. (2015)