Visual Tracking by Reinforced Decision Making

Janghoon Choi
ASRI, Seoul National University
ultio791@snu.ac.kr

Junseok Kwon
Chung-Ang University
jskwon@cau.ac.kr

Kyoung Mu Lee
ASRI, Seoul National University
kyoungmu@snu.ac.kr

Abstract

One of the major challenges of model-free visual tracking problem has been the difficulty originating from the unpredictable and drastic changes in the appearance of objects we target to track. Existing methods tackle this problem by updating the appearance model on-line in order to adapt to the changes in the appearance. Despite the success of these methods however, inaccurate and erroneous updates of the appearance model result in a tracker drift. In this paper, we introduce a novel visual tracking algorithm based on a template selection strategy constructed by deep reinforcement learning methods. The tracking algorithm utilizes this strategy to choose the best template for tracking a given frame. The template selection strategy is self-learned by utilizing a simple policy gradient method on numerous training episodes randomly generated from a tracking benchmark dataset. Our proposed reinforcement learning framework is generally applicable to other confidence map based tracking algorithms. The experiment shows that our tracking algorithm effectively decides the best template for visual tracking.

1. Introduction

Visual tracking is one of the most important and fundamental problems in the fields of computer vision and it has been utilized in many applications, such as automated surveillance, human computer interaction, and robotics. Also known as *model-free* object tracking, visual tracking algorithms aim to track an arbitrary object throughout a video segment, given the object's initial location as a bounding box representation.

For several years, visual tracking problem has been regarded as a *tracking-by-detection* problem, where the visual tracking task is formulated as an object detection task performed on consecutive video frames. Tracking algorithm is often composed of a combination of appearance and motion models of the object. Especially, the appearance model is carefully designed to be robust to numerous appearance variations of the target object, where common challenges arise from changes in illumination, motion blur, deformation and occlusion from surrounding objects [22].

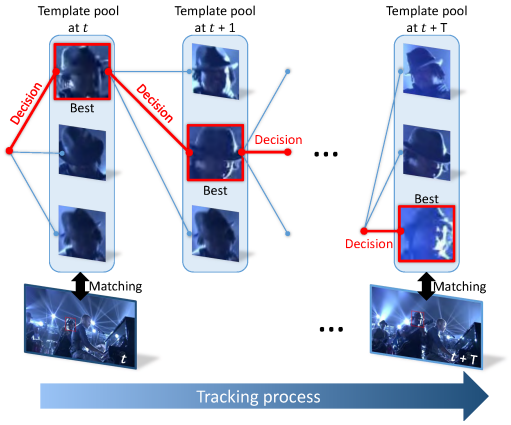


Figure 1. **Motivation for proposed visual tracking algorithm.** Our tracking algorithm formulates the visual tracking problem as a consecutive decision making task that can be self-learned through a reinforcement learning scheme. Our algorithm chooses the best template from a template pool to localize the target in a given frame.

To solve the aforementioned challenges, two approaches are mainly utilized to cover the appearance variations of the target object. One approach is to update the appearance model of the target on-line in the tracking process [17, 20, 32, 45, 14, 12], gaining new examples on the way. This approach considers the visual tracking problem as a semi-supervised learning task where the initial sample is labeled while other samples are not. However, inaccurate and erroneous update often causes the tracker to fail and drift to the background [24, 44]. The other approach is to utilize a feature representation scheme that is more robust to appearance perturbations while maintaining the discriminability between the target object and background objects [40, 39, 29, 38, 13]. This approach shares a common objective of other computer vision tasks such as object detection and semantic segmentation.

Recently, with growing attention on deep neural networks, especially convolutional neural networks (CNN)

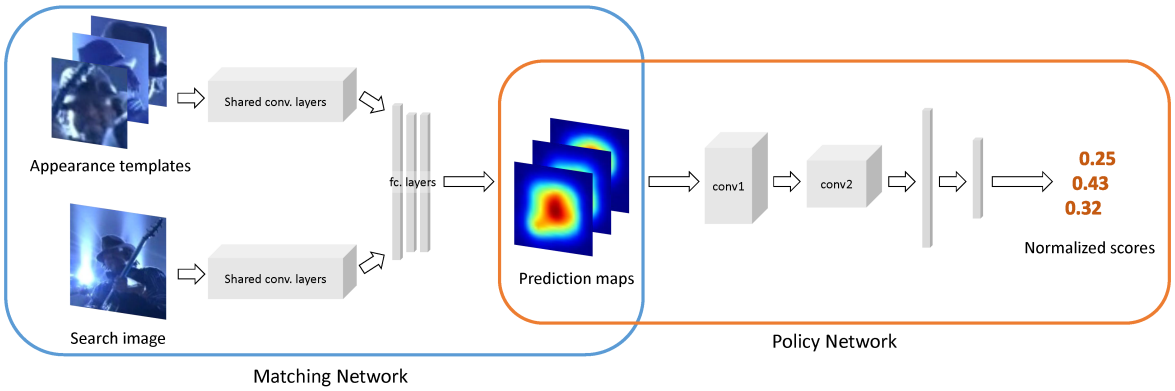


Figure 2. Overall architecture of the proposed system.

[21], there have been several approaches to utilize the powerful representation capabilities of CNNs for the visual tracking task. These methods showed successful results in covering the target appearance variations in short video segments. However, we also focus on the other aspect of visual tracking. Our proposed visual tracking algorithm aims to utilize the deep neural network for revising the on-line update by making decisions concerning which template is the most adequate for localizing the target in a new frame. Our method formulates the visual tracking task as a consecutive decision making process where given past target appearance samples, the tracker has to decide which sample is the best for localizing the target for a new frame. Figure 1 illustrates the motivation of this research.

While there are large image datasets [33] with ground truth labels available for obtaining a powerful feature representation under supervised learning environment, on-line update and selection of the target appearance model for visual tracking should be adequately tuned according to the capacity of the representation that is used. This results in an absence of explicit labels which should exist for a supervised learning environment. To resolve this problem, we adopt a reinforcement learning environment where given sequential states, an agent is prompted to make actions that can maximize the future reward. To achieve this learning task, we adopt deep neural networks for efficient state representation. Then we utilize policy gradient methods [41, 26] and experience replay memory [27], motivated by their recent success in playing the game of Go and ATARI video games [34, 28]. We train our policy network using randomly generated episodes from VOT-2015 tracking benchmark dataset [4]. We build our tracking algorithm based on a Siamese network [6] for its simplicity and fast tracking speed while having a powerful representation capacity. To our knowledge, our work is the first to utilize a deep reinforcement learning methodology for visual tracking.

2. Related Work

Conventional visual tracking algorithms can be largely categorized into two approaches. One approach constructs a generative model from previously observed samples and utilizes this model to find the region in the new frame where it can be described by the model best. The other uses a discriminative model where a classifier is learned to distinguish the target object region from the surrounding background region.

Generative approaches for visual tracking often utilize sparse representation [46, 45, 25] or linear subspace for incremental learning [20, 32]. Using these criteria, they try to find the target region where it can be described by the model. The model is constructed from target appearance samples collected from previously tracked frames. [32] uses principal component analysis on previous templates to construct a incremental subspace that can be used to reconstruct the target appearance. The target is localized by finding the location with the lowest reconstruction error. Discriminative approaches for visual tracking often utilize classifiers [2, 12, 17] or correlation filters [14, 23, 16, 9]. These approaches try to build a model that can distinguish the target appearance from the background region by using classification or regression. The model is trained from target and background appearance samples together. [12] uses structured SVM to find the transformation vectors for patches obtained from the vicinity of the target, solving the label ambiguity problem of the binary classification assumption. Other than the generative and discriminative methods, there are hybrid methods [47, 5] that aim to utilize the advantages of both models. [5] adopted two components for the appearance model; with one descriptive and the other discriminative. Both components are integrated through a single optimization task.

Recently, there have been approaches to utilize deep representations for the visual tracking task. Convolu-

tional neural networks (CNN) [21] have shown outstanding performance in a wide range of computer vision applications including image classification [19], object detection [31] and much more. Their powerful representation capacity motivated visual tracking approaches such as [40, 23, 39, 29, 38]. [40] was the first to introduce deep representation learning to visual tracking problem. They build a stacked denoising autoencoder and utilize its intermediate representation for visual tracking. In [23], hierarchical correlation filters learned on the feature maps of VGG-19 network [35] are efficiently integrated. [38] also utilizes the feature maps generated from the VGG network to obtain multi-level information. [29] used the structure of low-level kernels of VGG-M network [35] and trained on visual tracking datasets to obtain multi-domain representation for a robust target appearance model.

Based on deep representations, some outstanding performances were shown by using two-flow Siamese networks on stereo matching problem [43] and patch-based image matching problem [11]. Accordingly, approaches to solve the visual tracking problem as a patch matching problem have emerged [37, 3, 6, 13]. [37] and [13] train the Siamese networks using videos to learn a patch similarity matching function that shares an invariant representation. [3] and [6] further expand this notion and proposes a more end-to-end approach to similarity matching where a Siamese architecture can localize an exemplar patch inside a search image using shared convolutional layers. In particular, [3] proposes a fully-convolutional architecture that adopts a cross-correlation layer to obtain invariance to spatial transitions inside the search image, lowering the complexity of the training process significantly.

However, approaches [13, 6] use a naive on-line update strategy that cannot revise erroneous updates and recover from heavy occlusions. Moreover, approaches [37, 3] do not update the initial template, solely relying on the representation power of the pre-trained CNN. This approach may be effective for short-term video segments with no distractors, but the tracker can be attracted towards a distractor with a similar appearance to the target. Our proposed algorithm is aimed to solve both problems of the previous approaches, by utilizing previously seen examples to adapt to the recent appearance of the target and choosing the most adequate template for localizing the target, ruling out erroneously updated templates.

3. Proposed Algorithm

In the following subsections, we first show a brief overview of our proposed visual tracking algorithm. Then we describe the details of the proposed method. We show the theoretical background for the reinforcement learning formulation. Next we describe the architectures for visual tracking and its training scheme.

Algorithm 1: Visual tracking with reinforced decision making

input : Tracking sequence of length L
Initial target location x_0
output: Tracked target locations x_t

// For every frame in a sequence
for $t = 1$ *to* L **do**
  // For all N templates
  for $i = 1$ *to* N **do**
    Produce prediction maps s_t with each template i;
    Obtain normalized scores for each prediction map using policy network $\pi(a_t|s_t;\theta)$;
  end
  Choose the prediction map with maximum score;
  Localize the target x_t according to chosen prediction map;
  Add a template to template pool every K frames, discarding an old one;
end

3.1. Tracking with Reinforced Decisions

Our tracking system can be divided into two parts where the first part is the matching network that produces prediction heatmaps as a result of localizing the target templates inside a given search image. And the second part is the policy network that produces the normalized scores of prediction maps obtained from the matching network. Figure 2 shows the overall diagram of our tracking system.

Assuming the networks are trained and its weights are fixed, we can perform the visual tracking task on arbitrary sequences. For a given video frame, we crop and obtain a search image based on the target's previous bounding box information. Using the appearance templates obtained from previously tracked frames, the matching network produces prediction maps for each appearance template. Then the prediction maps are fed to the policy network where it produces the normalized scores for each prediction map. The prediction map with the maximum score is chosen and it is used to localize the target.

Appearance templates are obtained on a regular time interval during the tracking process. We intentionally use this simplistic template update method to promote robustness in decision making process. Overall flow of our tracking algorithm is described in algorithm 1.

3.2. Reinforcement Learning Overview

We consider a general reinforcement learning setting where the agent interacts with an environment through sequential states, actions and rewards. Given an environment $\mathcal{E}$ and its representative state s_t at time step t, an agent must

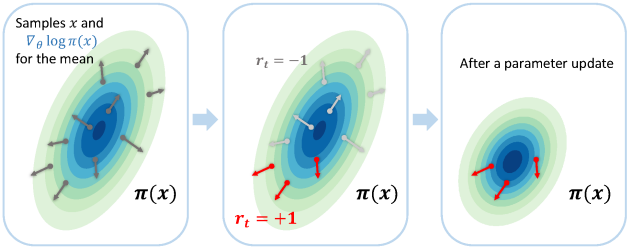


Figure 3. A conceptual visualization of policy refinement process.

perform an action a_t selected from a set $\mathcal{A}$ of every possible actions. Action a_t is determined by the policy $\pi(a_t|s_t)$ where action a_t can be chosen with deterministic or stochastic manner. In return for the action, the agent receives a scalar reward r_t and observes the next state s_{t+1}. This recurrent process continues until the agent reaches a terminal state. The goal of the agent is to select actions that maximizes the discounted sum of expected future rewards, where we define the action-value function as $Q^\pi(s,a)$.

To achieve the goal mentioned above, there are mainly two approaches to reinforcement learning; value-based methods and policy-based methods. Value-based method assumes that there exists an optimal action-value function $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ that gives the maximum action-value for a state-action pair, given some implicit policy (*e.g.* ϵ-greedy). The aim of value-based methods is to approximate the optimal action-value function using a function approximator as in $Q^*(s,a) \approx Q(s,a;\theta)$ where θ denotes the parameter for the function approximator. Minimization of the discrepancy between two functions can be achieved through variants of Q-learning algorithms [28].

On the other hand, policy-based method aims to directly model the policy function $\pi(a|s;\theta)$ without the assumption of intermediate action-value function, removing the need of evaluating the values of possible actions for a given state. Approximation of the policy function can be achieved by maximizing the objective return function R_t using stochastic gradient ascent algorithms. One simplest example of the method is the REINFORCE algorithm introduced in [41] where gradient ascent is used on expected reward $R_t = \mathbb{E}[r_t]$ as in

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi(a_t|s_t;\theta_{t-1}) r_t, \tag{1}$$

where α is the learning rate. Figure 3 shows a conceptual explanation for the policy refinement[1]. First, actions are sampled from the policy distribution π, then the performed actions are evaluated and rewards are given from interacting with the environment. Using this information, we can refine the policy through a parameter update. Samples from an updated policy is expected to give us a higher reward.

[1]Explanatory figure inspired by Andrej Karpathy's article, http://karpathy.github.io/2016/05/31/rl/

Algorithm 2: Training the policy network for a single episode

input : Randomly generated episode of length L, Policy network weights θ^-
output: Updated policy network weights θ^+

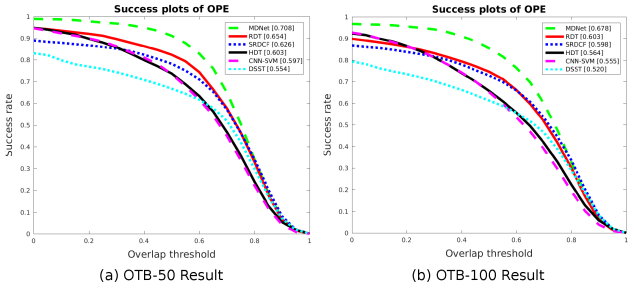
// For every frame in an episode
for $t = 1$ *to* L **do**
  // For all N templates
  for $i = 1$ *to* N **do**
    Produce prediction maps s_t with each template i;
    Obtain normalized scores for each prediction map using policy network $\pi(a_t|s_t;\theta^-)$;
  end
  Choose some a_t stochastically;
  Obtain gradient $\nabla_\theta \log \pi(a_t|s_t;\theta^-)$;
  Accumulate gradients according to eq. (2);
  Add a template to template pool every K frames, discarding an old one;
end
Sample from experience replay memory, obtain gradients;
if *episode successful* **then**
  Update weights $\theta^+ = \theta^- + \Delta\theta$
else
  Update weights $\theta^+ = \theta^- - \Delta\theta$
end

For our work, we use a variant of policy-based reinforcement learning method since it is commonly known to have better convergence properties and capability of learning stochastic policies [26]. In our reinforcement learning environment, we assume state s_t as the input prediction maps generated from the matching network and the action a_t is selecting the best prediction map for tracking a given frame. Reward r_t is given at the end of a tracking episode when the tracker successfully tracks the target, producing a bounding box overlap score over a predefined threshold.

3.3. Network Architectures

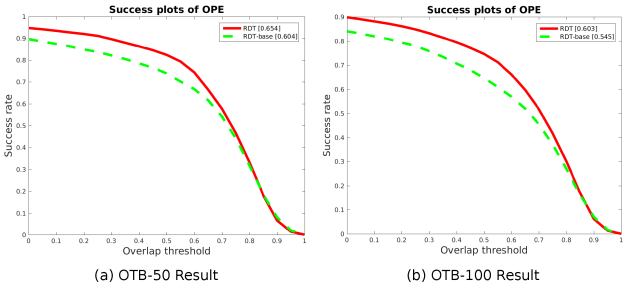
Architecture of the Matching Network: We borrow the Siamese architecture design from [6], using it as our baseline tracker. The Siamese architecture has 2 input branches and uses 3 shared convolutional layers for extracting the common representations from the object patch and the search patch. Then these features are concatenated and fed to 3 fully connected layers to produce a Gaussian prediction map, where the maximum point is the relative location of the object patch inside the search patch.

Architecture of the Policy Network: Our policy network sees the output prediction maps produced by the



(a) OTB-50 Result (b) OTB-100 Result

Figure 4. OPE result comparison on (a) OTB-50 and (b) OTB-100 benchmark dataset [42]. The numbers in the legend box indicate the average area-under-curve (AUC) scores for each tracker.



(a) OTB-50 Result (b) OTB-100 Result

Figure 5. OPE result comparisons of (a) OTB-50 and (b) OTB-100 benchmark dataset [42] for the internal comparisons.

matching network and makes the decision whether the matching result is reliable or not. By following this decision, we can always choose the best template for locating the target object in a given frame. The policy network consists of 2 convolutional layers to produce an adequate representation of the state and 2 fully connected layers for deciding whether this state is reliable or not. Then the outputs are fed through sigmoid function to produce probabilities. Finally, we choose the activation with the highest value and its corresponding template as the best candidate for tracking.

3.4. Training the Policy Network

To train the policy network $\pi(a|s;\theta)$ introduced above, we use a variant of REINFORCE algorithm with accumulated policy gradients. We randomly generate numerous tracking episodes with varying lengths from VOT-2015 [4] video dataset. Then we perform tracking on each training episodes with stochastically sampled action roll-outs produced by the policy network to ensure exploration of state space. For each decisions in an episode, we temporarily assume each decision was optimal and perform backpropagation to obtain gradients for all weights inside the policy network. We accumulate these gradients for all decisions in a single episode as in

$$\Delta\theta = \alpha \sum_{t=1}^{L} \nabla_\theta \log \pi(a_t|s_t;\theta)\beta^{L-t}, \qquad (2)$$

where L is the length of an episode, α is the learning rate and $\beta \in (0, 1]$ is a discounting factor inserted to give more weight to decisions made later in the episode. If an episode terminates, weights in the policy network are updated according to the success or failure of that episode. If an episode was successful, gradient is updated accordingly. If an episode was a failure, negative gradient is applied. The overall algorithm for training the policy network for a single episode is described in algorithm 2.

We also keep an experience replay memory of state-action-reward gained from previous episodes. When gradients are applied to the policy network after an episode, some amount of experiences are randomly sampled from the experience replay memory and applied concurrently. By using some sampled experiences from the experience replay memory, we can remove the correlation in incoming data sequence and reduce the variance of the update.

4. Experiments

4.1. Implementation Details

Policy network parameters: Input to the policy network are 31×31 prediction maps. The first convolutional layer has a 5×5 sized kernel with 4 output channels and it is applied with a stride of 3, then the activations are 2×2 max-pooled. The second convolutional layer has a 3×3 sized kernel with 8 output channels and it is applied with a stride of 1. Then the activations are fed to fully-connected layers, each with 128 hidden activations and 1 activation. For fully-connected layers, dropout regularization [36] is used with keep probability of 0.7. All layers are initialized from Gaussian normal distribution with zero mean and variance of 0.1 and each convolutional layer is followed by rectified linear unit (ReLU) activation functions.

Training parameters: To train the matching network, batch size of 64 is sampled from the ImageNet [33] dataset. For optimization, Adam optimizer [18] with learning rate of 10^{-4} is used. For the policy network, Adagrad optimizer [10] with learning rate of 10^{-4} is used and $\beta = 0.95$ is used. We train our policy network using 50,000 episodes randomly sampled from the VOT-2015 [4] benchmark dataset. Length of each episode is between 30 and 300 frames and a new template is added to the template pool every 50 frames. Success or failure of an episode is determined by the mean intersection-over-union (IoU) ratio of last 20 bounding box predictions compared to the ground truth bounding boxes. If the mean IoU is under 0.2, we consider the episode as a

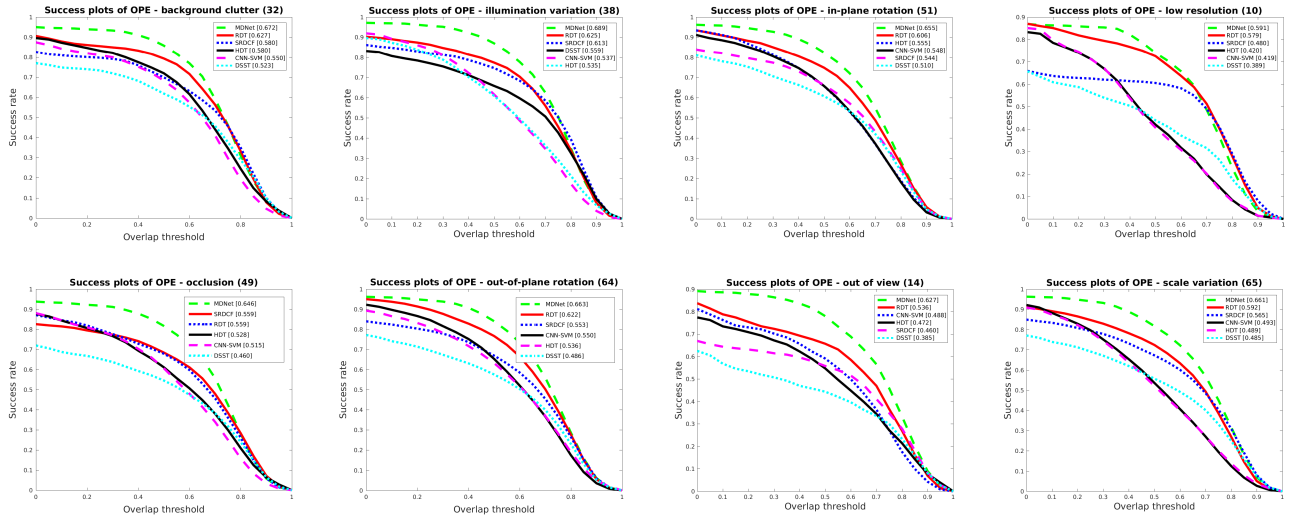


Figure 6. Success plots for 8 challenge attributes: background clutter, illumination variation, in-plane rotation, low resolution, occlusion, out-of-plane rotation, out of view and scale variation.

failure. To lower the variance of each update, we keep an experience replay memory for 5000 successful samples and 5000 failure samples. For each update, 40 samples are sampled from each experience replay memory and gradient is applied concurrently.

Tracking parameters: For practical reasons, we also average the location predictions obtained from 4 slightly shifted (upward, downward, left and right) search images to obtain a more accurate localization. And to cover the scale space, we use 3 scaled versions of the search image to find the best scale that fits the template. Scale parameters used are $1.05, 1.00$ and 1.05^{-1}. We used a maximum of 4 templates including the initial template for tracking, and template pool is updated every 50 frames, replacing an old template. Parameters were empirically selected to obtain the best performance achievable.

Implementation environment: We implement our tracker in Python using TensorFlow [1] library. The implementation runs on an Intel Core i7-4790K 4GHz CPU with 24GB of RAM and the neural network is computed and trained on GeForce GTX TITAN X GPU with 12GB of VRAM. Our implemented tracker runs at an average of 43 frames per second (FPS) on OTB-100 [42] video dataset.

4.2. Evaluation on OTB dataset

4.2.1 Quantitative Results

Object tracking benchmark (OTB) [42] is a well-known visual tracking benchmark dataset that contains 100 fully annotated sequences. We compare our tracking algorithm with 5 other state-of-the-art tracking algorithms published recently. MDNet [29], SRDCF [8], HDT [30], CNN-SVM [15] and DSST [7] tracking algorithms are used for comparison. Success plots for both OTB-50 and OTB-100 sequences are shown on figure 4 where the proposed algorithm is denoted as RDT. Success rate evaluation metric is calculated by comparing the predicted bounding boxes with the ground truth bounding boxes to obtain the IoU scores and measuring the proportion of scores larger than a given threshold value. Final score is calculated by measuring the area-under-curve (AUC) for each tracker. The result shows that our proposed algorithm achieves a competitive performance alongside with other trackers, despite using a simple patch matching framework. Our tracker also performs at a real-time speed of 43 FPS (Table 1), compared to other deep representation based trackers such as MDNet [29] running at 1 FPS and HDT [30] running at 10 FPS. The real-time processing speed of our tracker is a favorable characteristic for training the policy network using numerous training episodes.

Tracker	Environment	FPS
RDT	Python	43
MDNet [29]	MATLAB	1
CNN-SVM [15]	n/a	n/a
HDT [30]	MATLAB	10
SRDCF [8]	MATLAB	5
DSST [7]	MATLAB	24

Table 1. Tracking environment and speed comparison between trackers.

We further analyze the performance of our tracker for 8 different challenge attributes labeled for each sequence, where sequences with background clutter, illumination vari-

Figure 7. Qualitative results of the proposed method on challenging sequences from OTB benchmark dataset (in vertical order, *box, carScale, ironman, jump, matrix, singer1, soccer* and *bolt2*)

ation, in-plane rotation, low resolution, occlusion, out-of-plane rotation, out of view and scale variation are evaluated. As shown in figure 6, our tracker shows competitive results on most of the attributes compared to the other trackers.

To show the effectiveness of our reinforced template selection strategy, we also perform an internal comparison between two different versions of our algorithm. Figure 5 shows the comparison between two algorithms where RDT is the original algorithm and RDT-base is a variant where the template is selected at random. We were able to obtain a performance gain of roughly 10% for both OTB-50 and OTB-100 sequences by using the proposed template selection strategy, proving that our policy network chooses the more adequate template for tracking a given frame.

4.2.2 Qualitative Results

Figure 7 shows the snapshots of tracking results produced by the proposed algorithm with MDNet [29], SRDCF [8], HDT [30], CNN-SVM [15] and DSST [7]. Trackers were tested on some challenging OTB-100 sequences (*box, carScale, ironman, jump, matrix, singer1, soccer* and *bolt2*) where selected frame numbers are denoted as yellow on the top-left corners respectively. Our proposed tracking algorithm performs robustly, without losing track of the target under challenging conditions such as occlusion in *box* and *soccer*, scale change in *carScale*, *singer1* and *jump*, illumination variation in *ironman* and *matrix*. From the qualitative results, it is shown that our tracker successfully utilizes both the deep representation power and the template selection strategy for tracking the target.

For qualitative tracking results on more sequences, we attach a supplementary video for reference.

5. Conclusion

In this paper, we proposed a novel tracking algorithm based on a template selection strategy constructed by deep reinforcement learning methods, especially policy gradient methods. Our goal was to construct a policy network that can choose the best template from a template pool for tracking an arbitrary frame where the policy network is trained from numerous training episodes randomly generated from a tracking benchmark dataset. Experimental results show that we achieved a noteworthy performance gain in tracking under challenging scenarios, proving that our learned policy effectively chooses the appearance template that is more appropriate for a given tracking scenario.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 6

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. *arXiv preprint arXiv:1606.09549*, 2016. 3

[4] L. Čehovin, A. Leonardis, and M. Kristan. Visual object tracking performance measures revisited. *IEEE Transactions on Image Processing*, 25(3):1261–1274, 2016. 2, 5

[5] D. Chen, Z. Yuan, G. Hua, Y. Wu, and N. Zheng. Description-discrimination collaborative tracking. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014. 2

[6] K. Chen and W. Tao. Once for all: a two-flow convolutional neural network for visual tracking. *arXiv preprint arXiv:1604.07507*, 2016. 2, 3, 4

[7] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014. 6, 8

[8] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015. 6, 8

[9] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014. 2

[10] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. 5

[11] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015. 3

[12] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *Proceedings of the IEEE International Conference on Computer Vision)*, 2011. 1, 2

[13] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016. 1, 3

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 1, 2

[15] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *Proceedings of the 32nd International Conference on Machine Learning, 2015, Lille, France, 6-11 July 2015*, 2015. 6, 8

[16] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 749–758, 2015. 2

[17] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012. 1, 2

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3

[20] J. Kwon and K. M. Lee. Visual tracking decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 1, 2

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1, 3

[22] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. *ACM transactions on Intelligent Systems and Technology (TIST)*, 4(4):58, 2013. 1

[23] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision)*, 2015. 2, 3

[24] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):810–815, 2004. 1

[25] X. Mei and H. Ling. Robust visual tracking using l1 minimization. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1436–1443. IEEE, 2009. 2

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016. 2, 4

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 2013. 2

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2, 4

[29] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 3, 6, 8

[30] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, and J. L. M.-H. Yang. Hedged deep tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6, 8

[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 3

[32] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, pages 125–141, 2008. 1, 2

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 5

[34] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 2

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[36] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 5

[37] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[38] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 1, 3

[39] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 3

[40] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in neural information processing systems*, pages 809–817, 2013. 1, 3

[41] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 2, 4

[42] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 5, 6

[43] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *The Journal of Machine Learning Research*, 17(1):2287–2318, 2016. 3

[44] J. Zhang, S. Ma, and S. Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2014. 1

[45] T. Zhang, A. Bibi, and B. Ghanem. In defense of sparse tracking: Circulant sparse tracker. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[46] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[47] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1838–1845. IEEE, 2012. 2