POST: POlicy-based Switch Tracking

Ning Wang[1], **Wengang Zhou**[1], **Guojun Qi**[2], **Houqiang Li**[1]
[1]CAS Key Laboratory of GIPAS, University of Science and Technology of China
[2]Futurewei Technologies
wn6149@mail.ustc.edu.cn, guojun.qi@huawei.com, {zhwg, lihq}@ustc.edu.cn,

Abstract

In visual object tracking, by reasonably fusing multiple experts, ensemble framework typically achieves superior performance compared to the individual experts. However, the necessity of parallelly running all the experts in most existing ensemble frameworks heavily limits their efficiency. In this paper, we propose POST, a POlicy-based Switch Tracker for robust and efficient visual tracking. The proposed POST tracker consists of multiple weak but complementary experts (trackers) and adaptively assigns one suitable expert for tracking in each frame. By formulating this expert switch in consecutive frames as a decision-making problem, we learn an agent via reinforcement learning to directly decide which expert to handle the current frame without running others. In this way, the proposed POST tracker maintains the performance merit of multiple diverse models while favorably ensuring the tracking efficiency. Extensive ablation studies and experimental comparisons against state-of-the-art trackers on 5 prevalent benchmarks verify the effectiveness of the proposed method.

1 Introduction

Visual object tracking plays a vital role in many practical applications such as autonomous driving, video surveillance, human-computer interaction, etc. As a basic pre-processing component, visual tracking is also a time-critical task.

In the past decade, several popular tracking frameworks have emerged. Different trackers perform distinctively with their own strength and weakness. Compared with a single model that typically struggles to deal with various challenging factors, it is well recognized that ensemble-based approach, by absorbing the strength of multiple diverse experts/models, has the superiority to handle complex scenarios. In an ensemble framework, the core problem lies in how to effectively fuse or switch among multiple models. Some algorithms rely on the expert self-evaluation such as the output scores of different models (Han, Sim, and Adam 2017) or forward-backward trajectory consistency (Lee, Sim, and Kim 2015) to assess each single expert. On the other hand, some ensemble methods leverage the pair-wise relationship to vote on or evaluate the reliability of different experts (Wang and Yeung 2014). Despite their impressive performance, previous ensemble frameworks mostly suffer two limitations: (1) the fusion or selection strategies are manually designed in a heuristic manner and typically contain carefully selected hyperparameters; (2) the overall efficiency is inevitably reduced by running multiple models in parallel.

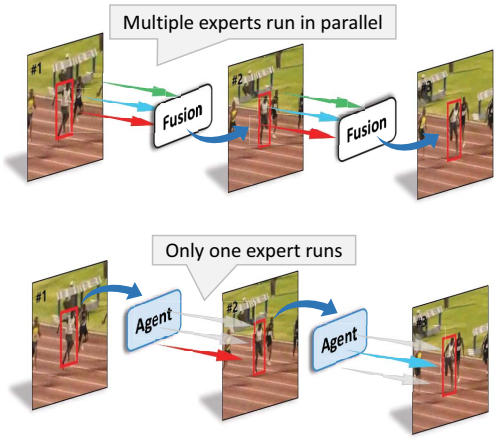


Figure 1: **Top**: classic ensemble trackers mostly operate multiple experts in parallel. **Bottom**: our method avoids unnecessary computations by directly assigning only one expert to handle the current frame.

When tracking objects, humans can rapidly capture the most discriminative features based on the tracking environment to distinguish the target from backgrounds. In different scenarios, our visual system takes advantage of different cues to track (e.g., color, shape, or semantics), and most of them may be redundant in a certain circumstance. The ensemble tracker shares partial similarity with the human visual system from the perspective of maintaining multiple cues for robust tracking. However, by concurrently operating multiple models, previous ensemble frameworks (Lee, Sim, and Kim 2015; Bailer, Pagani, and Stricker 2014; He et al. 2018; Han, Sim, and Adam 2017) generally make a tradeoff between efficiency and performance. So a question

emerges: can we avoid running all the models and directly find out a suitable one to still achieve high accuracy without sacrificing efficiency?

In this paper, we propose POST, a POlicy-based Switch Tracker for efficient and high-performance visual tracking. Similar to ensemble framework, our dynamic switch method maintains an ensemble of experts to retain the performance merit of diverse models. However, unlike the classic ensemble framework, our tracker is free of running multiple models online for result fusion, showing extremely promising efficiency. Specially, as shown in Figure 1, we only run one suitable expert in each frame, making our switch framework is almost as efficient as an individual expert and the overall computational cost is independent of the expert number. Such an adaptive switch relies on making the correct decision on selecting an expert in each frame. By virtue of Reinforcement Learning (RL) (Mnih et al. 2015), we learn an agent to predict the policy for the desired decision. The agent dynamically decides which is the suitable expert, and its decision-making capability is gradually improved by watching and exploring a large volume of videos offline following the RL rule. Different from previous ensemble frameworks putting emphasis on modeling the relationship of multiple experts, our agent predicts policies by directly analyzing the tracking environment as well as historic actions, which is more close to the judgment process of our visual system.

The main contribution of this work is the proposal of a policy-based switch tracking framework, which has been rarely investigated in the literature. Specifically, we build an expert ensemble using the state-of-the-art SiamRPN model (Li et al. 2018a), and combine this expert ensemble with an agent network to form our POST tracker. To encourage expert diversity and acquire superior accuracy, we further improve the base expert using an efficient color-histogram based model. Leveraging the agent trained via reinforcement learning, our framework dynamically switches different experts across frames, maintaining the performance advantage of diverse models while effectively avoiding unnecessary computations. The proposed method exhibits state-of-the-art performance on 5 prevalent tracking benchmarks while operating at a high speed of over 100 frames per second (FPS). Finally, it is worth mentioning that the proposed simple yet effective framework is quite generic, and has the potential for other tasks that benefit from such a multi-model selection or switch mechanism.

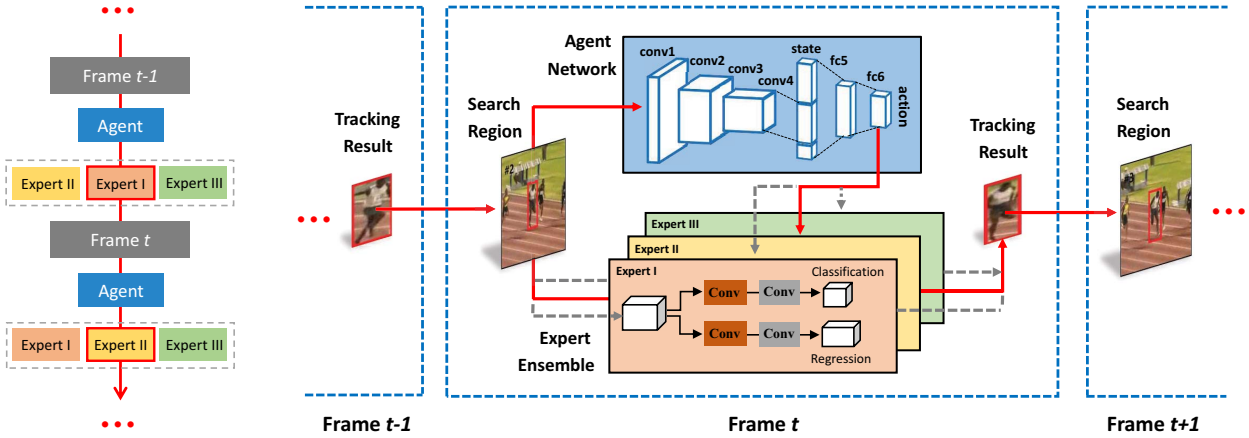
2 Related Work

In this section, we briefly review the related methods including visual object tracking, ensemble tracking and reinforcement learning based tracking approaches.

Visual Tracking. Visual object tracking aims to localize an unknown target object specified at the initial frame. In recent years, correlation filter based approaches (Henriques et al. 2015; Bertinetto et al. 2016a; Ma et al. 2015; Danelljan et al. 2016; 2017; Li et al. 2019) have attracted considerable attention, which tackle the tracking task by effectively solving a ridge regression problem in Fourier domain. By fusing multiple features (Ma et al. 2015; Qi et al. 2016; Danelljan et al. 2016), adaptive update (Huang and Zhou 2019), and boundary effect suppression (Galoogahi, Fagg, and Lucey 2017; Li et al. 2018b), the performance of correlation trackers have been greatly improved. The classification based methods (Nam and Han 2016; Song et al. 2018; Zhang, Huang, and Yang 2019) treat the tracking task as binary classification, which draw plentiful candidates in each frame and classify the target as well as background objects. The Siamese network based methods (Bertinetto et al. 2016b; Tao, Gavves, and Smeulders 2016) approach tracking task from the perspective of similarity estimation. On the basis of the Siamese framework, the correlation layer (Valmadre et al. 2017) and unsupervised learning (Wang et al. 2019) have been recently investigated. Since Siamese trackers are free of the online model update, they mostly exhibit real-time performance. The recent Siamese trackers with a region proposal network (Li et al. 2018a; Zhu et al. 2018) achieve a faster speed by discarding multi-scale estimation of the object.

Ensemble Tracking. To absorb the strength of different models, ensemble framework involves multiple weak experts to complement each other. The MEEM algorithm (Zhang, Ma, and Sclaroff 2014) exploits the relationship between the current tracker and its historical snapshots using entropy minimization. BranchOut method (Han, Sim, and Adam 2017) maintains multiple diverse fully-connected layers to better distinguish the target from backgrounds. The pair-wise relationship is also widely explored in vision tasks (Choi et al. 2016; Zeng et al. 2019). The multi-cue correlation tracker (Wang et al. 2018) considers both pair-wise relationship and self-consistency of multiple experts. In addition, some ensemble frameworks treat the off-the-shelf visual trackers as black boxes and analyze their returned bounding boxes to fuse the final results (Wang and Yeung 2014). However, the aforementioned fusion or selection strategies are mostly manually designed. By running multiple models in parallel, the efficiency is inevitably reduced. In contrast, as an alternative of the classic ensemble approaches, the proposed POST method is free of parallelly executing multiple experts and directly makes decisions by virtue of the agent trained via reinforcement learning.

RL in Visual Tracking. In recent years, RL has made impressive progress with the power of deep learning, and many algorithms have emerged in the RL field such as DQN (Mnih et al. 2015) and DDPG (Lillicrap et al. 2016). In the visual tracking community, many RL based trackers have been proposed. ADNet (Yun et al. 2017) learns policies to dynamically move and refine the bounding box for fast and accurate visual tracking. The Actor-Critic Tracker (ACT) (Chen et al. 2018) and DRL-IS approach (Ren et al. 2018) further exploit the continuous actions to tackle this issue. The EAly Stop Tracker (EAST) (Huang, Lucey, and Ramanan 2017) adaptively chooses cheap low-level or deep features via an agent. In (Dong et al. 2018), deep RL is used for hyperparameter optimization within a Siamese tracker. Different from the above methods, we leverage the RL technique from a different view, i.e., adaptively switch among different experts in each frame. Our simple yet effective RL based solution avoids the heuristic design of fusion rules as well as the unnecessary computational burden during online tracking.



(a) Framework Overview **(b) Policy-based Switch Tracker with Agent and Expert Networks**

Figure 2: The proposed switch framework is shown in (a), where the agent dynamically selects one expert to handle the current frame. The detailed POST tracker is shown in (b), which contains an ensemble of weak experts as well as an agent network.

3 Method

In Figure 2, we show an overview of the proposed policy-based switch tracker. The proposed POST method contains an agent and an expert ensemble. The expert ensemble is further composed of several weak experts/trackers with diverse tracking capabilities. In each frame, the agent receives the input environment and decides the action of choosing which expert. Note that only one expert is assigned to track in every frame and therefore the overall computational cost is independent of the expert number.

In the following, we first introduce the components of our expert ensemble in Section 3.1. Then, we present our agent model as well as the training details in Section 3.2.

3.1 Expert Ensemble

Revisiting SiamRPN. The Siamese network (Bertinetto et al. 2016b) comprises two identical branches to extract features from the template and search patches. Then the target feature representation is convolved with the feature of the search region to generate a response map. The maximum value of the response map represents the predicted target position. This process regards visual tracking as a similarity metric learning problem. The recent SiamRPN method (Li et al. 2018a) improves the SiamFC tracker (Bertinetto et al. 2016b) by incorporating an RPN model, which consists of two branches for classification and regression separately. It formulates the tracking task as one-shot object detection since the convolutional weights of classification and regression branches merely need to be computed once in the initial frame. During online tracking, the SiamRPN model directly detects the target in the search region, as shown in Figure 3. For more details, please refer to (Li et al. 2018a).

Improved SiamRPN. Siamese model utilizes the generated response map to distinguish the target from distractors. Similar objects may disturb the tracker and therefore a cosine window is applied to the response map to suppress the background distractors, which is considered as prior motion information. However, a simple cosine window fails to model the relationship between foreground and background objects since it merely penalizes the region far from the previous target location. In this work, we improve the robustness of Siamese trackers by incorporating a color-histogram based model (Possegger, Mauthner, and Bischof 2015). Specifically, we compute the color histograms of the target object $\mathcal{H}(\mathcal{O})$ and background region $\mathcal{H}(\mathcal{B})$. Then, for each pixel in a candidate patch (e.g., a pixel in k-th bin), we calculate its probability through $p^k = \frac{\mathcal{H}^k(\mathcal{O})}{\mathcal{H}^k(\mathcal{O})+\mathcal{H}^k(\mathcal{B})}$. The computed value $p^k \in [0, 1]$ denotes the probability of each pixel belonging to the target object. This color-histogram based per-pixel score map can better suppress the distractors as shown in Figure 4, and we combine it with the cosine window through the element-wise product to improve the tracking robustness.

Ensemble Details. We choose the DaSiamRPN (Zhu et al. 2018) as our base expert since (1) it achieves state-of-the-art performance on several standard benchmarks and outperforms its previous version (i.e., SiamRPN) due to superior model training strategy; (2) it is free of online model update, which enables us to safely switch among multiple experts in successive frames.

Different experts adopt diverse backbone networks or motion models. In our experiments, we construct 4 diverse experts: (1) Expert I adopts the standard DaSiamRPN model with an AlexNet-like backbone model; (2) Expert II is the improved Expert I with our aforementioned color model; (3) Expert III adopts a larger backbone network that doubles the model channel in the original DaSiamRPN; (4) Expert IV improves III via the proposed color model. In our experiments, we observe that different experts have distinctive tracking capabilities and their performance is variant in different videos, which means that they can complement each other to achieve superior accuracy. In the training stage, we first train the base experts following (Zhu et al. 2018). Then,

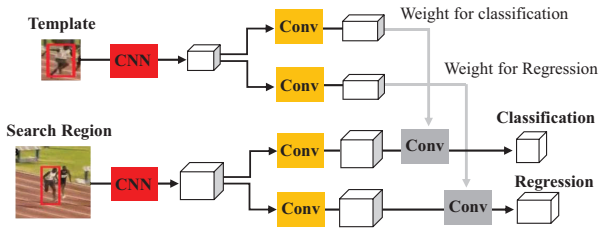


Figure 3: Architecture of the SiamRPN tracker.

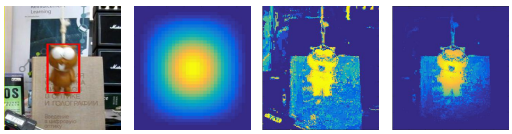


Figure 4: Given a search patch (a), previous Siamese trackers apply a cosine window (b) to the response map to penalize the surrounding background region. We improve this intuitive method by using an additional color-based per-pixel map (c). After combining (b) and (c), our method (d) can better suppress the distractors.

we fix their model parameters and further train the agent network via reinforcement learning.

3.2 Learning Policy via Reinforcement Learning

After constructing the expert ensemble, the key issue lies in how to assign a suitable expert in each frame. We formulate this expert switch problem as a Markov Decision Process (MDP). Formally, in MDP, there are a set of states S, actions A and a reward function R. In the t-th frame, the agent analyzes the current state S_t and takes an action A_t (i.e., expert selection in frame t). After taking the action, a scalar reward will be given reflecting how well the agent performs. By maximizing the total discounted sum of the future rewards, the agent learns the best policy to take actions (Mnih et al. 2015). In the following, we elaborate the state, action, reward and the training of our agent.

State. The state of our framework is a tuple $(f_{\text{init}}, f_{\text{cur}}, d_{\text{his}})$, where f_{init} is the feature representation of the search region in the initial frame, f_{cur} denotes the feature representation of the current search region, and d_{his} represents the vector of the history of taken actions. By involving the initial feature f_{init}, the agent receives a reference and learns the potential target and environment changes. This intuitive design shares partial similarity with the Siamese tracker, which proves to be effective in visual tracking.

In each frame, the search region image patch is resized to 107×107 and represented by a 512-dim feature vector via a CNN model with four convolutional layers. The history action vector d_{his} is obtained by concatenating the latest k actions. In our experiments, we maintain 4 experts and let $k = 5$, therefore the per-frame action is a 4-dim one-hot vector and d_{his} has 20 dimensions. The total state representation in each frame is $512 + 512 + 20 = 1044$ dimensional. As illustrated in Figure 2, after two fully-connected layers, this 1044-dim vector is mapped to a 4-dim vector, representing the expected future reward (i.e., Q value in DQN (Mnih et al. 2015)) of taking 4 different actions.

In our agent, the first four convolutional layers are initialized via the VGG-M network (Chatfield et al. 2014), and the following fully-connected layers are randomly initialized. Each layer is followed by a ReLU activation.

Action. As discussed above, the action of our agent is choosing an expert from the ensemble, which is represented by a 4-dim one-hot vector.

Given the current action in frame t, our POST method operates the corresponding expert to predict the tracking result. In frame $t+1$, we extract the search region based on the previous target position. Then the agent receives the next state S_{t+1} and will determine a new action A_{t+1}. By this state transition in successive frames, we achieve an MDP within the RL framework, as shown in Figure 2.

Reward. The reward function $R(S_t, A_t)$ reflects the performance influence by conducting the action A_t based on the state S_t. In visual tracking, previous RL based methods typically compute the overlap between the tracking result and ground-truth bounding box to give the reward. For instance, the reward is positive if the overlap score exceeds a certain threshold (e.g., 0.7), and vice versa (Yun et al. 2017; Chen et al. 2018). In our framework, since we aim to choose a suitable expert, we focus on *relative* performance gain instead of directly comparing with the ground-truth label. To better evaluate the tracking performance, we propose to simultaneously consider overlap precision (OP) and distance precision (DP), both of which are widely adopted evaluation metrics in visual tracking.

In the training stage, we operate all the experts. For i-th expert, we first compute the intersection-over-union (IoU) between its predicted bounding box B_i and the ground-truth G as follows:

$$O_i = \frac{\text{Area}(B_i \bigcap G)}{\text{Area}(B_i \bigcup G)}. \tag{1}$$

Then we calculate the center location error between B_i and the ground-truth: $D_i = \|\text{center}(B_i) - \text{center}(G)\|$, and further normalize it using Eq. 2.

$$\widetilde{D}_i = \exp\left(-\frac{1}{2\sigma^2} {D_i}^2\right), \tag{2}$$

where σ is the average value of the target width and height. By normalizing the center location error, $\widetilde{D}_i \in [0, 1]$ and can be combined with the overlap score O_i to comprehensively evaluate the tracking performance. Overlap metric and distance metric evaluate the performace from different perspectives. Therefore, unlike previous RL based trackers (Yun et al. 2017; Chen et al. 2018) merely utilizing OP, we propose to combine both OP and DP metrics. The final expert score P is defined as follows:

$$P_i = O_i \cdot \widetilde{D}_i. \tag{3}$$

Finally, we find out the highest expert score P_{max} among all the scores $\{P_i\}_{i=1}^{4}$. Our adaptive expert switch mechanism aims to improve the performance by choosing a better expert

in each frame. Therefore, supposing the action at state S_t is choosing the k-th expert, then the corresponding reward is given as follows:

$$R(S_t, A_t) = P_k - P_{\max}. \quad (4)$$

The agent will receive a higher reward if its chosen expert performs better among multiple experts, and the upper bound of the reward is 0 in case of the currently selected expert is optimal. Our designed reward function based on the relative performance gain forces our agent to predict better policies that bring performance improvement.

Training of Deep Q-Network. Since our action space is discrete and relatively low-dimensional, we utilize deep Q-learning (Mnih et al. 2015) to tackle this expert switch problem. Deep Q-Network (DQN) learns an action-value function $Q(S_t, A_t)$ to select the action A_t that provides the highest reward. In training, we could iteratively update the action-value function by

$$Q(S_t, A_t) = R_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}), \quad (5)$$

where γ is the discount factor. The function $Q(S, A)$ is learned via a deep Q-Network (i.e., the agent in Figure 2), which takes the state as input and returns a 4-dim Q-value vector. The action with the highest Q-value is regarded as the current optimal choice.

To iteratively update the Q function in Eq. 5, Deep Q-learning involves a target action-value function $\hat{Q}(S_t, A_t)$ with parameters $\theta^{'}$, which is copied from the Q function every τ steps. In training stage, given N pairs of (S_t, A_t, R_t, S_{t+1}) from the replay memory, the model $Q(S, A)$ can be learned by minimizing the following loss:

$$L = \frac{1}{N} \sum_t \left(y_t - Q(S_t, A_t; \theta)\right)^2, \quad (6)$$

where $y_t = R_t + \gamma \hat{Q}(S_{t+1}, A_{t+1}; \theta^{'})$. In addition, we also apply the ϵ-greedy strategy to gradually shift the model training from exploration to exploitation by gradually reducing the value of ϵ. Algorithm 1 summarizes the training process of our agent model.

Algorithm 1: Training of Deep Q-Network

Input: Training sequences and ground-truth.
Output: Trained weights for the Q network.

Initialize replay memory $\mathcal{M}$;
Initialize action-value function Q with weights θ;
Initialize target action-value function $\hat{Q}$ via $\theta^{'} = \theta$;
for *episode* $= 1, 2, ..., n$ **do**
  for $t = 1, 2, ..., t_{max}$ **do**
    if *random number* $\delta < \epsilon$ **then**
      select a random action A_t;
    else
      select the action A_t by Q network;
    end
    Handle the current frame using all the experts and compute their expert scores $\{P_i\}_{i=1}^4$;
    Choose the expert determined by the action and calculate the corresponding reward R_t;
    Obtain the next state S_{t+1} ;
    Store transition (S_t, A_t, R_t, S_{t+1}) in $\mathcal{M}$;
  end
  Sample a mini-batch of transitions (S_t, A_t, R_t, S_{t+1}) from $\mathcal{M}$;
  Update the network using Eq. 6;
  Update the $\hat{Q}$ by $\theta^{'} = \theta$ every τ steps;
  Reduce the ϵ-greedy threshold ϵ;
end

3.3 Tracking via POST

After offline training, our agent is readily ready for action prediction. Note that in online tracking, the agent fixes its pretrained parameters and does not receive any new reward.

The time cost of our agent is about 2.4 ms in each frame. Compared to the DaSiamRPN model, the computational cost of our lightweight agent is relatively small and therefore the action-making process merely slightly decelerates the overall speed. Thanks to our expert switch mechanism, our POST tracker achieves an average speed of multiple experts (about 105 FPS) and outperforms all of them in tracking accuracy.

4 Experiment

In this section, we first introduce the experimental details. Then, we analyze the effectiveness of our policy-based ensemble framework. Finally, we compare our method with state-of-the-art trackers on 5 standard tracking benchmarks including OTB-2013 (Wu, Lim, and Yang 2013), OTB-2015 (Wu, Lim, and Yang 2015), Temple-Color (Liang, Blasch, and Ling 2015), UAV123 (Mueller, Smith, and Ghanem 2016), and LaSOT (Fan et al. 2019).

4.1 Experimental Details

For the experiments on the OTB-2013, OTB-2015, Temple-Color, and UAV123, we use the videos from VOT-2013, VOT-2014, and VOT-2015 (Kristan et al. 2016) to train the agent and the overlapped videos are excluded. On the LaSOT dataset, we utilize its provided training videos to train the agent. We randomly choose continuous 20 to 40 frames in a video for each episode, and our agent is trained for 2×10^5 episodes using Adam optimizer. The size of the replay buffer $\mathcal{M}$ is set to 10000. The learning rate is 0.0001, discount factor $\gamma = 0.9$ and batch size is 128. As for the ϵ-greedy, we initially set $\epsilon = 1$ and reduce it by 5% every 2000 episodes and fix it after it is reduced to 0.1. The experimental environment is PyTorch on a computer with 4.00GHz Intel Core I7-4790K and NVIDIA GTX 1080Ti GPU.

In all experiments, we use one-pass evaluation (OPE) with distance and overlap precision metrics (Wu, Lim, and Yang 2013). The distance precision threshold is 20 pixels. The overlap success plot uses thresholds ranging from 0 to 1, and the area-under-curve (AUC) score is computed to evaluate the overall performance.

4.2 Framework Effectiveness Study

Effectiveness of Color-based Model. As discussed in Section 3.1, we improve the Expert I and III using a color-based

	Metric	I	II	III	IV	**POST (Ours)**
OTB-2013	DP	85.9	89.1	88.4	89.0	**90.7** (1.6∼4.8% ↑)
	AUC	65.3	66.0	65.9	66.4	**67.8** (1.4∼2.5% ↑)
OTB-2015	DP	86.2	86.8	85.8	87.6	**88.4** (0.8∼2.6% ↑)
	AUC	65.2	65.4	64.9	66.1	**67.2** (1.1∼2.3% ↑)
TC128	DP	74.6	76.4	73.4	75.1	**78.1** (1.7∼4.7% ↑)
	AUC	54.1	55.5	53.1	54.4	**56.3** (0.8∼3.2% ↑)
UAV123	DP	77.2	77.8	77.3	78.4	**80.0** (1.6∼2.8% ↑)
	AUC	59.3	59.4	61.2	61.5	**62.9** (1.4∼3.6% ↑)
LaSOT	DP	41.0	41.9	44.8	45.1	**46.3** (1.2∼5.3% ↑)
	AUC	43.2	44.0	46.8	47.2	**48.1** (0.9∼4.9% ↑)
Speed	FPS	160	145	94	82	105

Table 1: Comparison with our baselines (i.e., Expert I, II, III, and IV) on five challenging datasets. The evaluation metrics include distance precision at 20 pixels (DP score) and area-under-curve of the success plots (AUC score).

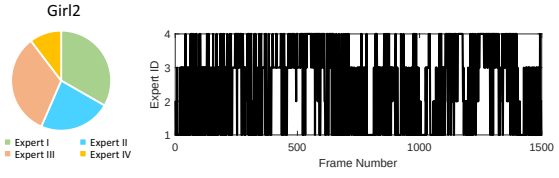


Figure 5: Expert running percentage (left) and the assgined expert per frame (right) in the *Girl2* video.

model. As shown in Table 1, Expert II outperforms I and Expert IV outperforms III, which illustrates the effectiveness of our method. The color model is robust at handling deformable targets (Bertinetto et al. 2016a) and can complement the template matching based Siamese model. It is worth mentioning that weak experts (e.g., Expert I) still play an important role since the model diversity is crucial for our switch framework.

Comparison with Base Experts. In Table 1, we compare the performance of our POST tracker with its individual experts. From the results, we can see that our final algorithm consistently outperforms its individual experts, ranging from 1.2%∼5.3% in DP metric and 0.8%∼4.9% in AUC metric on different benchmarks. In Figure 5, we visualize the expert per-frame switch in a video, where different experts are triggered in different scenes (or a short span of time).

As for the tracking speed, Expert I with a lightweight model exhibits the highest speed while its tracking performance is relatively unsatisfied. In contrast, Expert IV with a big backbone network and our color model achieves high accuracy while the tracking speed is the lowest. Our POST tracker significantly outperforms Expert I and II in tracking accuracy (about 3% in all datasets). Compared with the strong Expert III and IV, our POST tracker is still superior to them in *both* performance and *efficiency*. Overall, by dynamically switching different experts, our ensemble tracker achieves superior accuracy and a balanced speed.

Comparison with Other Fusion Strategies. We further compare our policy-based fusion method with manually designed strategies. In Table 2, the "Average" denotes the mean value of the bounding boxes of four experts. The "Max" represents choosing the expert with the highest classification score in each frame, which is widely adopted in ensemble based trackers. The "Random" denotes randomly choosing one expert in each frame. From the results in Table 2, we can observe that our POST method exhibits the best performance on all five datasets. Note that the "Average" and "Max" strategies need to run all the experts, and therefore the tracking speed is significantly reduced (only 32 FPS). In contrast, our method achieves a superior selection policy with promising tracking efficiency (105 FPS).

	OTB-2013	OTB-2015	TC-128	UAV123	LaSOT	FPS
Best	IV	IV	II	IV	IV	
Baseline	66.4	66.1	55.5	61.5	47.2	
Average	67.1	66.8	55.2	61.9	46.5	32
Max	66.6	66.4	55.6	61.5	47.0	32
Random	66.1	65.6	55.2	61.7	46.1	**128**
POST	**67.8**	**67.2**	**56.3**	**62.9**	**48.1**	105

Table 2: Comparison with manually designed fusion strategies. The evaluation metric is AUC score. Our framework achieves superior accuarcy while ensuring high efficiency.

4.3 Comparison with State-of-the-art Methods

In this section, we compare with state-of-the-art real-time trackers including KCF (Henriques et al. 2015), SiamFC (Bertinetto et al. 2016b), CFNet (Valmadre et al. 2017), SCT (Choi et al. 2016), CSR-DCF (Lukezic et al. 2017), Staple (Bertinetto et al. 2016a), ECOhc (Danelljan et al. 2017), ACT (Chen et al. 2018), ACFN (Choi et al. 2017), RT-MDNet (Jung et al. 2018), and DaSiamRPN (Zhu et al. 2018). In addition, we also compare with remarkable non-real-time trackers including HCF (Ma et al. 2015), ADNet (Yun et al. 2017), MDNet (Nam and Han 2016), C-COT (Danelljan et al. 2016), ECO (Danelljan et al. 2017), and VITAL (Song et al. 2018).

OTB-2013 and OTB-2015. OTB-2013 and OTB-2015 are widely used benchmarks in visual tracking with 50 and 100 videos, respectively. From Table 3, we can observe that our POST tracker achieves state-of-the-art performance on these two datasets. In Figure 6, we compare our POST tracker with some real-time trackers using precision and success plots. As illustrated in Figure 6, the proposed POST tracker obviously outperforms the recent real-time trackers.

The high performance of our POST method can be attributed to its strong baseline (i.e., DaSiamRPN) to some extent. However, compared with DaSiamRPN, our POST tracker still obviously outperforms it since (1) we improve the original SiamRPN method using a simple yet effective color-based model; (2) we further construct an ensemble of diverse SiamRPN models and dynamically switch among them using a pretrained agent.

Temple-Color. Temple-Color is a tracking dataset containing 128 color videos. On this benchmark, we also compare with the recent real-time and non-realtime trackers in Table 3. The proposed method achieves competitive performance among non-realtime trackers and outstanding performance

	Metric	KCF	SiamFC	Staple	ECOhc	RT-MDNet	DaSiamRPN	HCF	ADNet	MDNet	C-COT	ECO	VITAL	**POST**
OTB-2013	DP	74.0	80.9	79.3	87.4	-	85.9	89.0	90.3	94.8	89.9	93.0	**95.0**	90.7
	AUC	51.4	60.7	59.5	65.2	-	65.3	60.5	65.9	70.8	67.2	70.9	**71.0**	67.8
OTB-2015	DP	69.6	77.1	76.4	85.6	88.5	86.2	84.2	88.0	90.9	89.8	91.0	**91.7**	88.4
	AUC	48.5	58.2	58.1	64.3	65.0	65.2	56.6	64.6	67.7	67.1	**69.1**	68.2	67.2
TC-128	DP	54.9	68.8	67.1	75.1	78.8	74.6	-	-	-	78.1	**79.8**	-	78.1
	AUC	38.7	50.3	50.7	55.8	56.3	54.1	-	-	-	57.4	**59.7**	-	56.3
UAV123	DP	52.3	61.3	-	72.5	77.2	77.2	-	-	-	-	74.1	-	**80.0**
	AUC	33.1	39.9	-	50.6	52.8	59.3	-	-	-	51.7	52.5	-	**62.9**
LaSOT	DP	16.6	33.9	23.9	27.9	-	41.0	28.6	-	37.3	-	30.1	36.0	**46.3**
	AUC	17.8	33.6	24.3	30.4	-	43.2	25.0	-	39.7	-	32.4	39.0	**48.1**
Speed	FPS	**270**	83	70	50	46	160	12	3	1	0.3	6	1	105

Table 3: Comparison with state-of-the-art real-time (left) and non-realtime (right) trackers on five benchmarks including OTB-2013, OTB2015, Temple-Color, UAV123, and LaSOT. The evaluation metrics are distance precision at 20 pixels (DP score) and area-under-curve of the success plots (AUC score).

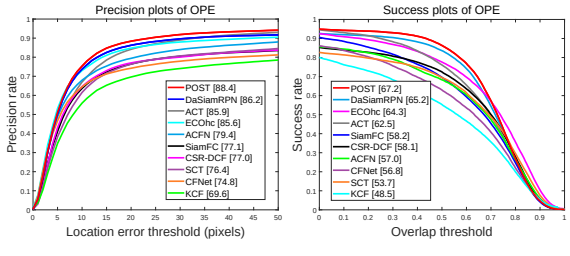


Figure 6: Precision and success plots on the OTB-2015 dataset for recent real-time trackers.

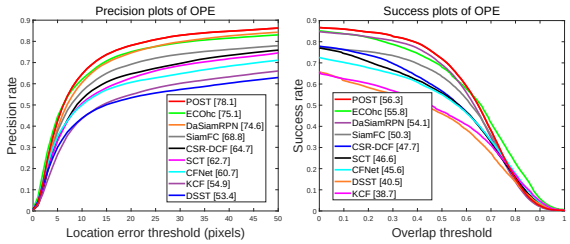


Figure 7: Precision and success plots on the Temple-Color dataset for recent real-time trackers.

among real-time trackers (Figure 7).

UAV123. In Table 3, we list the available results of some methods on the UAV123 dataset. Our POST tracker exhibits promising potential for low-altitude UAV tracking. For example, our approach significantly outperforms ECO by 5.9% DP score and 10.4% AUC score, respectively.

LaSOT. LaSOT is the recently released large-scale tracking dataset, which contains 1400 videos and the average video length is more than 2500 frames. As a consequence, LaSOT is much more challenging. In Figure 8, we compare the proposed POST tracker with top-ranked methods evaluated on the LaSOT using precision and success plots. As shown in Figure 8, on the testing set, our method surpasses other state-of-the-art trackers by a large margin. For example, the proposed POST tracker outperforms the best tracker in LaSOT (i.e., MDNet) by 9.0% DP score and 8.4% AUC score. Compared to our baseline DaSiamRPN, our method still outperforms it by 5.3% DP score and 5.1% AUC score as shown in Table 3.

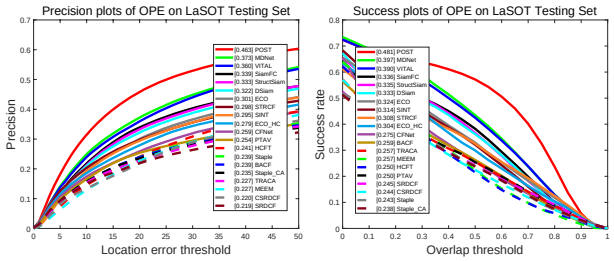


Figure 8: Evaluation on the LaSOT testing set using the precision and success plots. Only top 20 trackers are shown in the figure. Best viewed in color and zoom in.

5 Conclusion

In this paper, we propose a simple yet effective policy-based switch tracker, which contains an agent as well as an ensemble of Siamese models. In each frame, the agent assigns only one expert to track the target, making the computational complexity of our switch framework be $O(1)$ in terms of expert number. In addition, the making decision capability of our agent is offline learned following the RL rule, avoiding manually designing heuristic fusion strategies. Extensive experiments on five challenging benchmarks show that our POST tracker achieves state-of-the-art performance as well as satisfactory efficiency.

Acknowledgements. This work was supported in part to Dr. Wengang Zhou by NSFC under contract No.61632019 & No.61822208 and Youth Innovation Promotion Association CAS (No.2018497), and in part to Dr. Houqiang Li by NSFC under contract No.61836011.

References

Bailer, C.; Pagani, A.; and Stricker, D. 2014. A superior tracking approach: Building a strong tracker through fusion. In *ECCV*.

Bertinetto, L.; Valmadre, J.; Golodetz, S.; Miksik, O.; and Torr, P. 2016a. Staple: Complementary learners for real-time tracking. In *CVPR*.

Bertinetto, L.; Valmadre, J.; Henriques, J. F.; Vedaldi, A.; and Torr, P. H. 2016b. Fully-convolutional siamese networks for object tracking. In *ECCV*.

Chatfield, K.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*.

Chen, B.; Wang, D.; Li, P.; Wang, S.; and Lu, H. 2018. Real-time 'actor-critic' tracking. In *ECCV*.

Choi, J.; Jin Chang, H.; Jeong, J.; Demiris, Y.; and Young Choi, J. 2016. Visual tracking using attention-modulated disintegration and integration. In *CVPR*.

Choi, J.; Jin Chang, H.; Yun, S.; Fischer, T.; Demiris, Y.; and Young Choi, J. 2017. Attentional correlation filter network for adaptive visual tracking. In *CVPR*.

Danelljan, M.; Robinson, A.; Khan, F. S.; and Felsberg, M. 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*.

Danelljan, M.; Bhat, G.; Shahbaz Khan, F.; and Felsberg, M. 2017. Eco: Efficient convolution operators for tracking. In *CVPR*.

Dong, X.; Shen, J.; Wang, W.; Liu, Y.; Shao, L.; and Porikli, F. 2018. Hyperparameter optimization for tracking with continuous deep q-learning. In *CVPR*.

Fan, H.; Lin, L.; Yang, F.; Chu, P.; Deng, G.; Yu, S.; Bai, H.; Xu, Y.; Liao, C.; and Ling, H. 2019. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*.

Galoogahi, H. K.; Fagg, A.; and Lucey, S. 2017. Learning background-aware correlation filters for visual tracking. In *ICCV*.

Han, B.; Sim, J.; and Adam, H. 2017. Branchout: Regularization for online ensemble tracking with convolutional neural networks. In *CVPR*.

He, A.; Luo, C.; Tian, X.; and Zeng, W. 2018. A twofold siamese network for real-time object tracking. In *CVPR*.

Henriques, J. F.; Caseiro, R.; Martins, P.; and Batista, J. 2015. High-speed tracking with kernelized correlation filters. *TPAMI* 37(3):583–596.

Huang, J., and Zhou, W. 2019. Re2ema: Regularized and reinitialized exponential moving average for target model update in object tracking. In *AAAI*.

Huang, C.; Lucey, S.; and Ramanan, D. 2017. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*.

Jung, I.; Son, J.; Baek, M.; and Han, B. 2018. Real-time mdnet. In *ECCV*.

Kristan, M.; Matas, J.; Leonardis, A.; Vojíř, T.; Pflugfelder, R.; Fernandez, G.; Nebehay, G.; Porikli, F.; and Čehovin, L. 2016. A novel performance evaluation methodology for single-target trackers. *TPAMI* 38(11):2137–2155.

Lee, D. Y.; Sim, J. Y.; and Kim, C. S. 2015. Multihypothesis trajectory analysis for robust visual tracking. In *CVPR*.

Li, B.; Yan, J.; Wu, W.; Zhu, Z.; and Hu, X. 2018a. High performance visual tracking with siamese region proposal network. In *CVPR*.

Li, F.; Tian, C.; Zuo, W.; Zhang, L.; and Yang, M.-H. 2018b. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*.

Li, Y.; Zhu, J.; Hoi, S. C.; Song, W.; Wang, Z.; and Liu, H. 2019. Robust estimation of similarity transformation for visual object tracking. In *AAAI*.

Liang, P.; Blasch, E.; and Ling, H. 2015. Encoding color information for visual tracking: algorithms and benchmark. *TIP* 24(12):5630–5644.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *ICLR*.

Lukezic, A.; Vojir, T.; Cehovin Zajc, L.; Matas, J.; and Kristan, M. 2017. Discriminative correlation filter with channel and spatial reliability. In *CVPR*.

Ma, C.; Huang, J.-B.; Yang, X.; and Yang, M.-H. 2015. Hierarchical convolutional features for visual tracking. In *ICCV*.

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

Mueller, M.; Smith, N.; and Ghanem, B. 2016. A benchmark and simulator for uav tracking. In *ECCV*.

Nam, H., and Han, B. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*.

Possegger, H.; Mauthner, T.; and Bischof, H. 2015. In defense of color-based model-free tracking. In *CVPR*.

Qi, Y.; Zhang, S.; Qin, L.; Yao, H.; Huang, Q.; and Yang, J. L. M.-H. 2016. Hedged deep tracking. In *CVPR*.

Ren, L.; Yuan, X.; Lu, J.; Yang, M.; and Zhou, J. 2018. Deep reinforcement learning with iterative shift for visual tracking. In *ECCV*.

Song, Y.; Ma, C.; Wu, X.; Gong, L.; Bao, L.; Zuo, W.; Shen, C.; Lau, R. W.; and Yang, M.-H. 2018. Vital: Visual tracking via adversarial learning. In *CVPR*.

Tao, R.; Gavves, E.; and Smeulders, A. W. 2016. Siamese instance search for tracking. In *CVPR*.

Valmadre, J.; Bertinetto, L.; Henriques, J. F.; Vedaldi, A.; and Torr, P. H. 2017. End-to-end representation learning for correlation filter based tracking. In *CVPR*.

Wang, N., and Yeung, D. Y. 2014. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *ICML*.

Wang, N.; Zhou, W.; Tian, Q.; Hong, R.; Wang, M.; and Li, H. 2018. Multi-cue correlation filters for robust visual tracking. In *CVPR*.

Wang, N.; Song, Y.; Ma, C.; Zhou, W.; Liu, W.; and Li, H. 2019. Unsupervised deep tracking. In *CVPR*.

Wu, Y.; Lim, J.; and Yang, M.-H. 2013. Online object tracking: A benchmark. In *CVPR*.

Wu, Y.; Lim, J.; and Yang, M.-H. 2015. Object tracking benchmark. *TPAMI* 37(9):1834–1848.

Yun, S.; Choi, J.; Yoo, Y.; Yun, K.; and Young Choi, J. 2017. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*.

Zeng, R.; Huang, W.; Tan, M.; Rong, Y.; Zhao, P.; Huang, J.; and Gan, C. 2019. Graph convolutional networks for temporal action localization. In *ICCV*.

Zhang, Y. Q. S. Z. W.; Huang, L. S. Q.; and Yang, M.-H. 2019. Learning attribute-specific representations for visual tracking. In *AAAI*.

Zhang, J.; Ma, S.; and Sclaroff, S. 2014. Meem: robust tracking via multiple experts using entropy minimization. In *ECCV*.

Zhu, Z.; Wang, Q.; Li, B.; Wu, W.; Yan, J.; and Hu, W. 2018. Distractor-aware siamese networks for visual object tracking. In *ECCV*.