Online Object Tracking Based on CNN with Metropolis-Hasting Re-Sampling

Xiangzeng Zhou and Lei Xie*
School of Computer Science
Northwestern Polythechnical University
Xi'an, P. R. China
xenuts@gmail.com, lxie@nwpu.edu.cn

Peng Zhang* and Yanning Zhang
School of Computer Science
Northwestern Polythechnical University
Xi'an, P. R. China
{zh0036ng, ynzhang}@nwpu.edu.cn

ABSTRACT

Tracking-by-learning strategies have been effective in solving many challenging problems in visual tracking, in which the learning sample generation and labeling play important roles for final performance. Since the concern of deep learning based approaches has shown an impressive performance in different vision tasks, how to properly apply the learning model, such as CNN, to an online tracking framework is still challenging. In this paper, to overcome the overfitting problem caused by straight-forward incorporation, we propose an online tracking framework by constructing a CNN based adaptive appearance model to generate more reliable training data over time. With a reformative Metropolis-Hastings re-sampling scheme to reshape particles for a better state posterior representation during online learning, the proposed tracking outperforms most of the state-of-art trackers on challenging benchmark video sequences.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Tracking*

General Terms

Algorithm, Theory

Keywords

Object tracking, CNN, Metropolis-Hastings, Re-sampling

1. INTRODUCTION

Learning sample quality is an essential factor to robust online tracking, but this task is not easy because it is hard to manually intervene the sample generation and labeling when tracking is on-the-fly. Although different tracking strategies have tried various types of traditional models for sample generation[15], the descriptive capability of those online sample is still far from sufficient for object characteristic representation.

In order to exploit more descriptive training samples, nowadays, deep learning models, e.g. convolutional neural network (CNN) [16], have been successfully applied in a variety of audio and visual tasks such as speech recognition and image classification, and obtain a remarkable progress. But due to the requirement of a large number of training data and high computational cost, most of those studies approached their tasks with off-line learning process as presented in some recently proposed works [14, 11, 7]. Wang *et al.* [14] proposed an online tracking strategy based on a compact image representation learned from an off-line pretrained deep neural network which requires large amounts of auxiliary images. Similarly, Hong *et al.* [7] carried out the learning of discriminative saliency using a CNN, but still demanded a pre-trained model. Different with [14] and [7], Li *et al.* [11] proposed a variation of CNN with truncated structural loss to construct an online tracker and showed promising performance. But it mainly focused on model reforming of CNN for online learning, and the sample generation problem is not addressed, which may lead to tracking failure in complicated scenarios. Thus, how to utilize the advantage of deep learning to generate more representative samples is a challenging problem in online tracking tasks, and this is also a motivation of this study.

Sample labeling is another challenge to properly utilize a CNN model as learning strategy for online tracking. This is because CNN is prone to overfitting to recent samples and is sensitive to mislabeled samples. Typically, a particle filter is used for efficiently conducting online object tracking by simulating object state's posterior with a finite set of weighted particles. However, it is difficult for the particles being used to carry out a self-repair without any prior knowledge when a specific error pattern arises. Such type of error may be caused by an incorrect object's interference due to dramatic appearance change or overfitting problem (e.g. CNN based appearance model). Therefore, an effective re-sampling process over particle filter may benefit the label assignment, providing more reliable labeled samples for the learning of CNN model. This is another motivation leading to this study.

In this work, we propose a robust online tracker by exploiting the strong learning capability of a CNN model with particle filtering framework. An overview of our tracking framework is shown in Fig. 1. The contributions of the proposed work are three folds. Firstly, we carry out an attempt by introducing a single convolutional neural network

*Corresponding author.

MM'15, October 26–30, 2015, Brisbane, Australia.

DOI: http://dx.doi.org/10.1145/2733373.2806307.

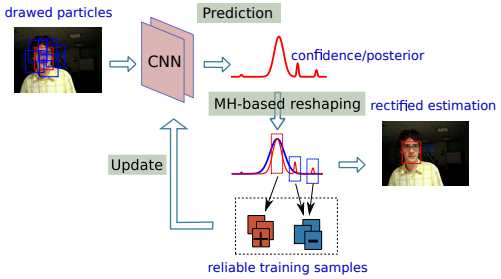

Figure 1: Our online tracking framework

for object appearance modeling and integrate it into particle filtering framework with online fashion. Secondly, a novel re-sampling scheme is proposed over particles based on the Metropolis-Hastings (MH) algorithm [5, 2] to obtain a set of more reliable positive and negative samples, which could conduct a more effective and robust training of CNN based appearance model. Thirdly, the proposed re-sampling method provides a way to apply a heuristic prior to reshape the drawn particles and gains a better object state posterior representation. Additionally, we apply an adaptive weighted approach with samples to feed CNN and a "lazy" training style is used.

The organization of the paper is as follows: Section 2 presents the single CNN architecture used in the proposed tracking framework. Section 3 and Section 4 elaborate our proposed Metropolis-Hastings based re-sampling method with an online robust tracking using CNN. Section 5 shows experimental demonstration and Section 6 draws conclusions.

2. APPEARANCE MODELING WITH CNN

In the proposed tracking framework (Fig. 1), we construct a single convolutional neural network (CNN) shown in Fig. 2, as an appearance model for object description, which consists of two convolutional layers with tanh as activation function and followed by max pooling operators. Then, there is a fully connected layer followed by a softmax classifier. The working mechanism of this model is: the input data is generated from gray-scale image and is locally normalized with $\sigma = 5$. Then, the image patches are normalized to size 40×40. The first convolution layer contains 10 kernels with size 3×3 followed by a max-pooling layer with size 2×2. The second convolution layer contains 20 kernels with size 3×3 as well as followed by a 2×2 max-pooling operation. The fully connected layer with size 500 is followed by a softmax classifier with cross-entropy loss function to produce confidence for each input patch.

The CNN is initially trained well in the first frame and is updated in a lazy manner only when the confidence of estimated object is lower than an empirical threshold $\mathcal{C} = 0.9$. In the updating phase, the training goal is to advance the confidence of each positive and negative sample above $\mathcal{C}$. Our CNN model allows to accept weighted training samples

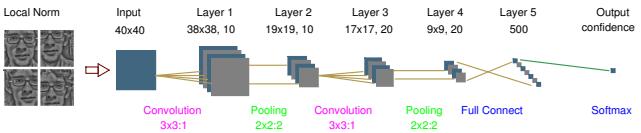

Figure 2: CNN architecture

as inputs. Inspired by the spirit of boosting, the weight of each sample is estimated according to the objective loss in former iteration. The process is beneficial to promote CNN's updating speed.

3. PARTICLE FILTERING WITH METROPOLIS-HASTINGS RE-SAMPLING

Particle filter has been widely used to conduct online tracking within a Bayesian framework of which a prediction of state $\mathbf{x}$ in time t is formulated as:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}. \quad (1)$$

The key of particle filter is efficiently propagating the posterior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ using a finite set of N particles $\{\mathbf{x}_t^i\}$ attached with importance weights w_t^i that can be estimated as:

$$w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^i) \cdot p(\mathbf{x}_t^i \mid \mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})}. \quad (2)$$

And importance distribution $q(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$ is often simplified to $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ with first-order Markov assumption.

Unfortunately, the particle filter framework does not supply a mechanism in nature to self-repair the mode drifting incurred by the accumulative error of observation model $p(\mathbf{y}_t \mid \mathbf{x}_t^i)$. The accumulative error is often caused by dramatic appearance change or the model's over-fitting problem. In practice, a heuristic assumption holds: the probability distribution of object's state should be a unimodal Gaussian distribution $\mathcal{P}(\mathbf{x} \mid \mathbf{y}) = \mathcal{N}(\mu, \sigma^2)$, of which we can take advantage to rectify the particles. Inspired by this, we propose a re-sampling method to reshape particles $\{w_t^i,\ \mathbf{x}_t^i\}$ using a variation of Metropolis-Hastings algorithm.

3.1 Reshape Particles by Metropolis-Hastings

The Metropolis-Hastings algorithm [5, 2] is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. A sketch of Metropolis-Hastings algorithm is shown in Algorithm 1.

Algorithm 1 Metropolis-Hastings Algorithm

1: Randomly choose a starting value $\mathbf{x}^0$.
2: At iteration m, draw a candidate $\hat{\mathbf{x}}$ from a proposal distribution $J(\hat{\mathbf{x}} \mid \mathbf{x}^{m-1})$.
3: Compute an acceptance ratio α (shown in Fig. 3).
4: Accept $\hat{\mathbf{x}}$ as $\mathbf{x}^m$ with a probability $\min(\alpha, 1)$, otherwise $\mathbf{x}^m = \mathbf{x}^{m-1}$.
5: Repeat step 2-4 until M samples are got.

The core step of the Metropolis-Hastings algorithm is to compute an *Acceptance Ratio* for m-th iteration as a fashion in Eq. 3.

$$\alpha = \frac{p(\hat{\mathbf{x}} \mid \mathbf{y}) \cdot J(\mathbf{x}^{m-1} \mid \hat{\mathbf{x}})}{p(\mathbf{x}^{m-1} \mid \mathbf{y}) \cdot J(\hat{\mathbf{x}} \mid \mathbf{x}^{m-1})} \quad (3)$$

In the m-th iteration, a sample candidate $\hat{\mathbf{x}}$ is generated by a proposal distribution $J(\hat{\mathbf{x}} \mid \mathbf{x}^{m-1})$. The sample candidate $\hat{\mathbf{x}}$ will be accepted as the m-th sample $\mathbf{x}^m$ with a probability $\min(\alpha, 1)$. In this work, the probability distribution we need to simulate is exactly the state posterior $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.

For each time t, we conduct an M iterations Metropolis-Hastings sampling process over particles $\{w_t^i,\ \mathbf{x}_t^i\}$ by computing a new *Acceptance Ratio* as:

$$\alpha = \frac{\mathcal{P}(\hat{\mathbf{x}}_t \mid \mathbf{y}_{1:t}) \cdot J(\mathbf{x}_t^{m-1} \mid \hat{\mathbf{x}}_t)}{p(\mathbf{x}_t^{m-1} \mid \mathbf{y}_{1:t}) \cdot J(\hat{\mathbf{x}}_t \mid \mathbf{x}_t^{m-1})}, \quad (4)$$

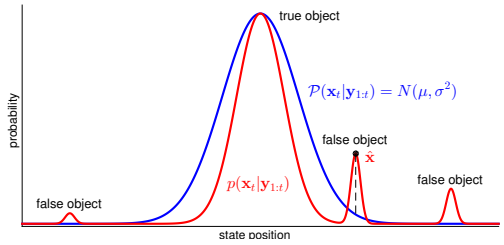

Figure 3: Reshaping posterior distribution

where $\mathcal{P}(\hat{\mathbf{x}}_t \mid \mathbf{y}_{1:t})$ is a Gaussian distribution that we want the particles to approach as far as possible, and $\hat{\mathbf{x}}_t$ is a sample candidate proposed by the proposal distribution $J(\hat{\mathbf{x}}_t \mid \mathbf{x}_t^{m-1})$ in time t. Fig. 3 shows us an illustration of this method in 2D space. For each sample candidate $\hat{\mathbf{x}}$, when it is situated in the neighborhood of the false object with a high probability, it will get a small acceptance ratio α. For another case, while a sample candidate is situated in the neighborhood of the true object, the value of α shall approach to 1, which means a high probability to be accepted.

Throughout this process, the higher the probability of particles belonging to a false object, the more heavily they will be penalized via acceptance ratio. The shape of rectified particles are more close to real posterior. Apart from this, the acceptance ratio α brings an extra profit. Particles with largest α are suggested as reliable positive samples, and those with lowest α then should be taken as negative samples which exactly identify some specific error patterns. Training samples obtained through this process put more focus on several notable error patterns and produce a more robust appearance model.

4. APPROXIMATE ESTIMATION

In the previous section, we present a Metropolis-Hastings based re-sampling method over particles to reshape a better posterior, and suggest a set of representative training samples at the same time. A remaining major problem is how to estimate the unknown target Gaussian distribution $\mathcal{P}(\mathbf{x} \mid \mathbf{y})$. In this section, we propose an approximate estimation algorithm, elaborated in Algorithm 2, to efficiently estimate the target distribution over time. We leave out

Algorithm 2 top-down Breadth-first Segmentation

Init: $\mathcal{Q} \leftarrow \emptyset$, $F^{1:N} \leftarrow 0$, $flag \leftarrow 1$, $K \leftarrow 1$
while $\exists F^i = 0$ **do**
  $\mathbf{x}^i \leftarrow \exists i, \forall j, C^i \geqslant C^j \wedge F^i = 0 \wedge F^j = 0$
  $F^i \leftarrow flag$
  ENQUEUE($\mathcal{Q}$, $\mathbf{x}^i$)
  while $\mathcal{Q} \neq \emptyset$ **do**
    $\hat{\mathbf{x}} \leftarrow$ DEQUEUE($\mathcal{Q}$)
    $\mathcal{X}_\epsilon = \{\mathbf{x}^k \mid \forall k, \|\hat{\mathbf{x}} - \mathbf{x}^k\|_2 \leq \epsilon \wedge C^k \leq \hat{C}\}$
    $F_\epsilon \leftarrow flag$
    ENQUEUE($\mathcal{Q}$, $\mathcal{X}_\epsilon$)
  end while
  $Seg_K = \{\mathbf{x}^k \mid \forall k, F^k = flag\}$
  $K \leftarrow K + 1$, $flag \leftarrow flag + 1$
end while

the time subscript for the sake of brevity. At time t, Algorithm 2 is conducted over N particles $\{\mathbf{x}^i\}$ associated with confidence C^i estimated by a CNN based appearance model. $\mathcal{Q}$ is a First-Come-First-Served(FCFS) queue and F is a flag vector of length N, each of whose position is initialized as 0. This algorithm starts from an untagged particle $\hat{\mathbf{x}}$ with global maximum confidence and performs a top-down FCFS tagging process. In a ϵ-neighborhood of $\hat{\mathbf{x}}$, only particles with lower confidence C^k are accepted to put into queue $\mathcal{Q}$ and tagged with the same flag as $\hat{\mathbf{x}}$. Until the queue is empty, a segmentation of particles will be available by tracing the flag. The algorithm ends up with no more untagged particles left and K segments are eventually obtained. Please note that these segments may intersect with each other because no constraint of $F^k = 0$ is required in step 8. As shown in Fig. 4, a diagrammatic example of segmentation result is provided for better understanding. The right panel of Fig. 4 gives an insight on the result of segmentation from 3D viewpoint.

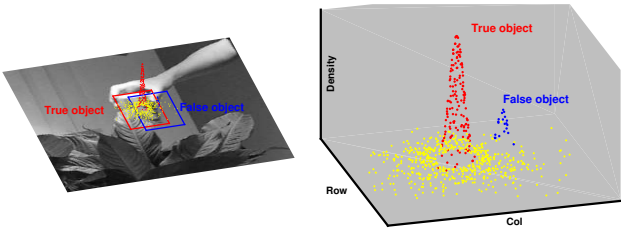

Figure 4: The top-down segmentation algorithm

Throughout this top-down segmentation process, we can easily achieve a rectified estimation of object's state in time t as:

$$\mathbf{x}_t^{est} = \arg\max_{\mathbf{x}^{k,i}} \max_{c^{k,i}} \{c^{k,i} \times Sup(Seg_k)\},\ k \in [1, K] \qquad (5)$$

where $Sup(Seg_k)$ is the size of support set of k-th segmentation. This estimation is more robust than a typical estimation method that selects the particle with highest confidence. After that, the support set of the selected segmentation can be utilized to estimate the target Gaussian distribution $\mathcal{P}(\mathbf{x} \mid \mathbf{y})$.

5. EXPERIMENTS

Experiment settings: We evaluate our proposed tracker with 20 video sequences from an online tracking benchmark [15] which covers most challenging tracking scenarios such as scale and illumination change, occlusion, fast movement, out of plane rotation and background clutters. We compare our method with 11 state-of-the-art tracking approaches: CSK [6], CT [17], CXT [3], IVT [13], LSK [12], MIL [1], SCM [18], Struck [4], TLD [8], VTD [9] and VTS [10]. The tracking performances are evaluated via average VOR (Pascal VOC overlap ratio) and average CLE (center location error) [15].

Comparison results: The elaborated quantitative results of VOR and CLE are shown in Table. 1, in which the Top-3 results are highlighted in red, blue and cyan, respectively. The overall satisfactory performance has confirmed that the proposed method can achieve more robust tracking in a variety of challenging scenarios with an average VOR of 95.67%. In addition, success-rate curves and precision curves are shown in Fig. 5 to demonstrate the general performance of all the tracking methods. We have also conducted the experiments based on CNN without Metropolis-Hastings (MH) re-sampling process and the obtained VOR has decreased to 66.47% in average, which demonstrated an

essential role of the MH re-sampling in our tracking framework. The proposed work is mainly implemented by MATLAB, and the average speed is 2-3 fps without any code optimization.

Table 1: Comparison of 12 trackers on 20 video sequences. Upper panel: VOR, lower panel: CLE.

	Ours	CSK	CT	CXT	IVT	LSK	MIL	SCM	TLD	VTD	VTS	Struck
boy	99.3	84.2	68.8	49.7	32.6	*99.7*	38.5	43.9	93.5	78.6	79.6	*97.5*
car4	100	27.6	27.5	29.9	*100*	5.61	27.6	*97.3*	*78.1*	35.4	35.2	39.8
carDark	98.2	*99.2*	0.25	69.0	69.7	*100*	17.8	*99.7*	52.9	68.4	*100*	*100*
coke	85.9	*73.9*	9.28	59.1	13.1	16.2	11.7	33.7	28.9	13.7	14.4	*94.2*
crossing	100	31.7	*98.3*	34.2	24.2	11.7	*98.3*	*100*	51.7	41.7	40.0	*94.2*
david	92.4	23.6	42.7	83.4	79.4	58.6	22.9	*91.3*	*97.0*	67.7	73.0	23.6
deer	100	*100*	4.23	*91.5*	2.82	33.8	12.7	2.82	*73.2*	4.23	4.23	*100*
dudek	94.3	94.7	85.2	92.4	96.8	92.7	85.7	97.6	84.2	*100*	*99.6*	*98.0*
faceocc1	100	*100*	85.4	77.1	*97.5*	40.8	76.5	*100*	83.4	*92.5*	88.3	*100*
fish	100	4.20	88.9	*100*	*100*	33.2	38.7	86.3	*96.2*	64.3	*97.9*	*100*
girl	96.8	39.8	17.8	64.2	18.6	34.4	29.4	*88.2*	76.4	65.2	52.6	*98.0*
jumping	99.4	4.79	0.64	28.8	9.90	6.39	47.6	12.1	*84.7*	11.2	15.7	*79.9*
mhyang	100	*100*	73.0	*100*	*100*	*100*	38.9	*99.7*	89.3	94.8	*97.0*	*100*
mntBike	100	*100*	17.1	28.1	*98.2*	90.4	57.5	96.1	25.9	*100*	*99.6*	85.5
singer1	100	29.6	24.8	32.2	*48.1*	19.7	27.6	*100*	*99.1*	43.0	42.5	29.9
singer2	87.7	3.55	1.09	3.83	3.83	4.10	*47.5*	16.4	3.01	*45.1*	39.1	3.55
sylvester	83.6	71.7	82.8	74.7	67.6	26.3	54.6	*88.6*	*92.8*	80.4	80.7	*92.9*
trellis	95.3	59.1	35.0	80.8	30.9	*90.7*	24.4	*85.4*	47.3	50.1	49.0	78.4
walking2	100	38.8	38.4	39.8	*100*	*48.6*	38.0	*100*	34.0	40.2	40.4	*43.4*
woman	80.6	24.5	15.9	20.6	18.4	18.9	18.8	*85.8*	16.6	18.1	17.1	*93.5*
boy	2.23	20.1	9.03	7.39	91.3	*2.24*	12.8	51.0	4.49	7.57	7.27	*3.84*
car4	2.84	19.1	86.0	58.1	*2.15*	66.6	50.8	*4.27*	13.6	37.0	36.7	8.69
carDark	1.17	3.23	119	16.5	8.43	*1.29*	43.5	1.30	27.5	16.5	2.87	*0.95*
coke	14.4	*13.6*	40.5	25.7	83.0	55.0	46.7	56.8	25.1	68.7	62.5	*12.1*
crossing	1.76	8.96	3.56	23.4	*2.36*	54.8	3.18	*1.57*	24.3	26.1	43.1	2.81
david	5.64	17.7	10.5	6.05	*4.82*	12.0	16.9	*4.34*	*5.12*	11.6	10.2	42.8
deer	4.56	*4.97*	246	6.75	183	98.8	101	104	6.26	135	220	*5.27*
dudek	14.7	13.4	26.5	12.8	*9.62*	14.6	17.7	10.8	18.1	*10.3*	*9.85*	11.4
faceocc1	12.7	*11.9*	25.8	25.3	18.4	30.4	29.9	*13.0*	27.4	20.2	21.3	18.8
fish	4.19	41.2	10.7	6.25	*5.67*	50.8	24.1	8.54	6.57	16.8	7.21	*3.40*
girl	3.68	19.3	18.9	11.0	22.5	29.3	13.7	*2.60*	9.79	8.60	13.0	*2.57*
jumping	3.94	86.0	47.7	9.99	61.6	74.6	9.99	65.9	*5.94*	41.4	40.1	*6.55*
mhyang	2.39	3.61	13.3	3.97	*1.87*	3.43	20.4	*2.41*	9.51	4.36	4.07	2.59
mntBike	7.72	*6.51*	214	179	*7.66*	11.5	73.0	10.6	209	9.78	9.67	8.63
singer1	4.32	14.0	15.5	11.4	11.3	20.8	16.4	*2.72*	7.99	*4.19*	5.35	14.5
singer2	10.3	185	127	164	175	149	*22.5*	114	*8.47*	43.7	72.5	174
sylvester	13.2	9.92	8.56	14.8	34.2	68.4	15.2	*7.97*	*7.31*	19.6	19.4	*6.30*
trellis	3.73	18.8	41.7	7.01	120	*4.70*	71.5	7.01	31.6	32.3	24.3	*6.92*
walking2	3.35	17.9	58.5	34.7	*2.46*	18.9	60.6	*1.65*	52.5	46.2	54.0	11.2
woman	9.64	207	114	72.5	177	131	125	*7.88*	93.9	119	121	*4.17*

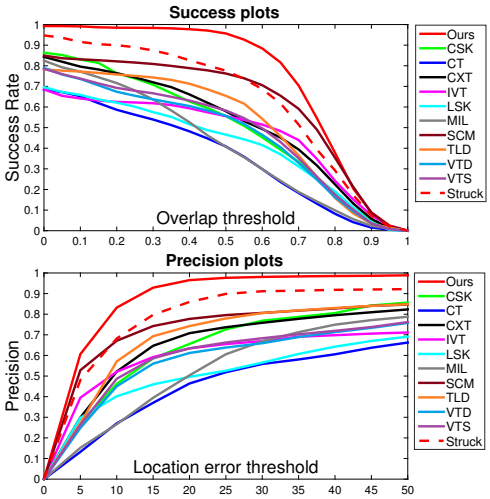

Figure 5: The success plots and precision plots

6. CONCLUSIONS

In this paper, we propose a robust online tracking approach for general objects with a convolutional neural network (CNN) based appearance model. To alleviate the problems incurred by direct use of CNN for online tracking task, we present a re-sampling method over particles with a variation of Metropolis-Hastings algorithm to gain better posteriors, and draw a set of more reliable training samples to feed CNN at the same time. With this proposed method, our tracker can employ CNN for online robust tracking task effectively and efficiently. Compared with several state-of-art trackers, the proposed tracker shows more satisfactory performance in various challenging scenarios.

7. ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (61175018, 61301194, 61231016), Research Fund for the Doctoral Program of Higher Education of China (20126102120055), foundation grant from NWPU (3102014JSJ0014).

8. REFERENCES

[1] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, pages 983–990, June 2009.

[2] R. Bardenet, A. Doucet, and C. Holmes. Towards scaling up markov chain monte carlo: an adaptive subsampling approach. In *ICML*, pages 405–413, 2014.

[3] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, pages 1177–1184, 2011.

[4] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, Nov 2011.

[5] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[6] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012.

[7] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. *arXiv:1502.06796*, 2015.

[8] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, pages 49–56, June 2010.

[9] J. Kwon and K.-M. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[10] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *ICCV*, pages 1195–1202, 2011.

[11] H. Li, Y. Li, and F. Porikli. Robust online visual tracking with a single convolutional neural network. In *ACCV*, pages 194–209, 2014.

[12] B. Liu, J. Huang, C. Kulikowski, and L. Yang. Robust visual tracking using local sparse appearance model and k-selection. *PAMI*, 35(12):2968–2981, Dec 2013.

[13] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[14] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013.

[15] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2015.

[16] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer International Publishing, 2014.

[17] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, pages 864–877, 2012.

[18] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845, June 2012.