DEEP CONVOLUTIONAL FEATURE HISTOGRAMS FOR VISUAL OBJECT TRACKING

Paraskevi Nousi *Anastasios Tefas* *Ioannis Pitas*

Department of Informatics, Aristotle University of Thessaloniki, Greece

ABSTRACT

Visual Object Tracking remains an open and challenging task in the Computer Vision field, requiring tracking algorithms to achieve a feeble balance between precision and speed performance. In this work, inspired by the classic Mean Shift algorithm for object tracking using histograms, as well as the recent advances of deep Convolutional Neural Networks (CNNs), we propose a novel tracker that incorporates elements from both worlds. Our tracker uses a deep CNN as the feature extraction backbone, which is capable of extracting semantically meaningful features from the target and its background, as well as a fully learnable Bag-of-Features mechanism which extracts histograms from those features. The tracker operates in a fully-convolutional fashion, allowing for the direct and efficient evaluation of multiple possible target locations. Extensive experimental results demonstrate the efficiency and effectiveness of the proposed tracker, allowing it to run at high speeds even on systems with lower computational capacity.

***Index Terms*—** Visual Object Tracking, Convolutional Neural Networks, Convolutional Bag of Features

1. INTRODUCTION

In the field of computer vision, Visual Object Tracking refers to the localization of an arbitrary object of interest over subsequent video frames, given its initial appearance and location. A tracker will typically extract and maintain a model of the object and use that model to search for and locate the target in successive frames, the challenge lying in avoiding similar objects, while dealing with appearance changes, occlusions, cluttered backgrounds, fast movement, etc. In real-life applications, visual analysis algorithms are also required to run at real-time speed [1], and tracking algorithms should be equipped to deal with tracking an object over several lost frames, which is significantly more challenging or even prohibitive depending on the tracker's speed.

In a typical tracking scenario, the tracker first builds a target model, which should be discriminative enough that the tracker doesn't drift, albeit forgiving to appearance changes. An intuitive approach to such problems is to update the target model with each subsequent frame, in a way such that small changes are incorporated into the model gradually. Depending on the nature of the tracker however, this process may cause significant delays to the tracking process. To accommodate real-time applications, trackers must be efficient in both their localization strategy as well as their model update procedure.

Amongst the most pioneering and foundational tracking algorithms lie Discriminative Correlation Filters (DCF) [2], and Mean Shift (MS) [3, 4], tracking approaches. More recent trackers exploit the semantically meaningful feature representations extracted by Convolutional Neural Networks (CNNs) and typically rely either on online optimization of the network parameters or offline learning to produce discriminative features while maintaining high tracking speed.

In this work, we revisit the histogram-based tracking paradigm utilized by MS-based trackers while making use of CNNs for feature extraction to benefit from their success. We employ a fully learnable Bag-of-Features module into the feature extracting process of a CNN to extract histograms from both the target and a wider search area. Using these histograms, which encode convolutional feature map information into compact representations, we are able to locate the target by matching the target and area histograms in a fully convolutional fashion. The proposed tracker runs at high speeds even on devices with limited computational capabilities while achieving significant results on multiple tracking benchmarks.

The rest of this paper is organized as follows. Section 2 is a summary of related work on visual object tracking. Section 3 introduces and analyzes in depth the proposed tracker and methodology. The experimental results, which validate our hypotheses, are presented in Section 4, and finally Section 5 summarizes our findings.

2. RELATED WORK

Recently, convolutional neural networks have met great successes in various visual analysis fields, including object recognition, detection and tracking. Trackers based on convolutional neural networks, have recently started to attract research attention. Trackers like MDNet [5], ECO [6], have achieved state-of-the-art performance on various object tracking benchmarks, at the cost of very slow tracking speeds. We focus on more lightweight trackers which can run at real-time speeds.

GOTURN [7], is a CNN-based tracker, trained to regress the new target position, given the target and a candidate region. The target and candidate regions are first mapped to an intermediate representation using a fully convolutional network. The two representations are then fed to a series of fully connected layers, and the final layer predicts the new position and size of the target. Amongst its drawbacks are the need for severe data augmentation during training, to simulate as many changes in position and scale as possible, as well as the tracker's performance.

SiamFC [8] is another CNN-based tracker, trained as a fully convolutional siamese network, which performs cross correlation between the features extracted from the target and a candidate region to find the new position of the target. In contrast to GOTURN, data augmentation is unnecessary, due to the network's fully convolutional nature. Its successor, CFNet [9], included a Discriminative Correlation Filter (DCF) module, presented as a fully learnable layer. Ever since, more approaches have combined DCFs with CNNs, such as DCFNet [10].

The classic Mean Shift (MS) and its adaptive version [4, 3] both use color histograms extracted from the target weighted by a differentiable kernel profile, to finally extract the new position of the target using the mean shift vector. In this work, histograms of convolutional features are extracted from both the target and candidate regions, using a fully learnable Bag-of-Features module, and compared using histogram matching measures in a convolutional fashion. The new position of the target is chosen by finding the position at which the maximum similarity is achieved. The use of histograms makes such tracking algorithms robust to appearance changes, such as pose and viewpoint [11].

3. PROPOSED METHOD

3.1. Histogram-based Tracking

In classic Mean Shift tracking [4], the target is represented by a spatially weighted histogram of features extracted from each spatial location of the input image corresponding to the target. Let an input image be represented by features $\mathbf{f} \in \mathbb{R}^{H\times W\times C}$, lying on a regular lattice $\mathbf{x} \in \mathbb{R}^{H\times W\times 2}$, where C is the number of channels and H, W are the spatial dimensions. Also let $\mathbf{f}^{\star}_{ij} \in \mathbb{R}^{h\times w\times C}$ represent those features that spatially correspond to the target, with h, w being its spatial dimensions, and $\mathbf{x}^{\star}_{ij} \in \mathbb{R}^{h\times w\times 2}$ represent its spatial coordinates, centered around $\mathbf{0}$, with i, j indexing the coordinates corresponding to the target. Finally, let M denote the number of histogram bins extracted from the features by some quantization method. The target model is then represented as an M-dimensional histogram $\mathbf{q} \in \mathbb{R}^{M}$, acquired by spatially weighing the histograms corresponding to the target using a Gaussian kernel profile $k(\cdot)$:

$$q_m = C \sum_{i,j} k(\|\mathbf{x}^{\star}_{ij}\|^2)\delta[b(\mathbf{f}^{\star}_{ij}) - m)] \tag{1}$$

where $C = 1/\sum_{ij} k(\|\mathbf{x}^{\star}_{ij}\|^2)$, and $b(\cdot) \in 1, \ldots, M$ maps its argument to a histogram bin index.

Respectively, the *target candidate model* representation at spatial location $\mathbf{y}$ is denoted as $\mathbf{p}(\mathbf{y})$ and formulated as:

$$p_m(\mathbf{y}) = C_h \sum_{ij} k(\|\frac{\mathbf{y} - \mathbf{x}_{ij}}{h}\|^2)\delta[b(\mathbf{f}_{ij}) - m] \tag{2}$$

where $C_h = 1/\sum_{ij} k(\|\frac{\mathbf{y}-\mathbf{x}_{ij}}{\sigma}\|^2)$ is a normalization constant arising from the condition that $\sum_{m=1}^{M} p_m(\mathbf{y}) = 1$, σ is the kernel radius and i, j index the coordinates corresponding to the last known position of the target. The similarity between the resulting histogram representations can be measured using the Bhattacharyya coefficient, which ultimately leads to the Mean Shift update vector to find the target's new position.

3.2. Convolutional Bag-of-Features based Tracking

Similar models can be obtained using a Bag-of-Features (BoF) based quantization method on features extracted by a Convolutional Neural Network [12]. Let $\mathbf{c} \in \mathbb{R}^{M\times C}$ be M codeword feature vectors, one for each of the M bins of the histograms. The feature vectors $\mathbf{f} \in \mathbb{R}^{H\times W\times C}$ can be quantized into histogram bins by assigning each feature vector to the bin with whose codeword it is most similar. To avoid harsh binary representations, the similarity between each feature vector and each codeword can be used instead, provided that the histograms are subsequently normalized to sum up to 1. Thus, memberships $\mathbf{g} \in \mathbb{R}^{H\times W\times M}$ are created, indicating the similarity of each feature vector in $\mathbf{f}$ with each of the codewords $\mathbf{c}$. Let $\mathbf{h} \in \mathbb{R}^{h\times w\times M}$ denote these histogram representations, whose spatial dimensions in the general case are different than those of the original feature vectors. For example, from an original input volume one histogram can be extracted by averaging the memberships at all spatial locations. In the case of Mean Shift, the binary memberships are spatially weighted by a Gaussian kernel.

Thus, three main components are required for the extraction of BoF histogram representations. First, a similarity measure to extract the memberships of each feature vector in the input volume with each of the codewords. Secondly, a membership normalization method, such as l_1 normalization. And finally, a membership averaging method, to produce the final histogram which should combine the memberships from all spatial locations. In terms of convolutional neural networks this can be translated into first extracting memberships using a Radial Basis Function (RBF) layer, which first computes the memberships as the Euclidean similarity of each input feature vector with each of the codewords and subsequently normalizes these memberships by their sum over all codewords. Formally, let $g_{ij,m}$ denote the membership of the feature vector at spatial location $\{i, j\}$ to the m-th codeword, then its memberships are computed as:

$$g_{ij,m} = \frac{\exp(-\|\mathbf{f}_{ij} - \mathbf{c}_m\|_2^2)}{\sum_{n=1}^{M} \exp(-\|\mathbf{f}_{ij} - \mathbf{c}_n\|_2^2)} \tag{3}$$

Alternatively, the similarity between each input feature vector to each codeword may be measured as their dot-product (or cosine similarity if they are l_1-normalized). This directly translates into using a traditional convolutional layer where the filters act as codewords. The similarity between all codewords and all input vectors is computed by the layer's operation if the filters are not flipped (i.e., cross-correlation), as is typical. In this case the memberships can be computed as:

$$g_{ij,m} = \frac{|\mathbf{c}_m^T \cdot \mathbf{f}_{ij}|}{\sum_{n=1}^{N} |\mathbf{c}_n^T \cdot \mathbf{f}_{ij}|} \tag{4}$$

where the absolute value of the dot product is used to enforce similarity metric properties for Equation (4).

Finally, histograms may be extracted by averaging over the spatial locations of the computed memberships:

$$\mathbf{h} = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{g}_{ij} \tag{5}$$

This is translated to a global average pooling operation in terms of neural network components. The histogram extraction process is illustrated in Figure 1. To maintain spatial

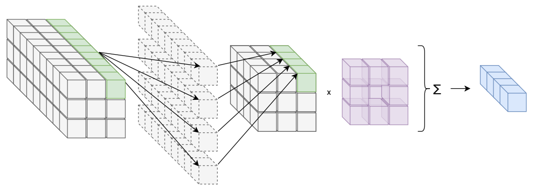


Fig. 1: Summarization of the histogram extraction process. Each feature vector in the original input volume of size $3 \times 3 \times 9$ is compared against each of the 4×9 codewords, and memberships of size $3 \times 3 \times 4$ are extracted. After an average pooling operation of size 3×3, a single histogram of size $1 \times 1 \times 4$ is extracted.

information, which is crucial in tracking tasks, multiple histograms can be extracted by averaging the memberships over local regions. For example, four histograms can be extracted by dividing the spatial locations into a 2×2 pooling grid. This can be achieved by using (strided) local pooling layers. Depending on the size of the pooling grid and the stride, this process can approximate the spatial gaussian weighting used in MS, as feature vectors lying on the edges will contribute to fewer histograms than feature vectors lying in more central positions.

The more histograms used, the more spatial information is maintained but more comparisons are required between target histograms and candidate histograms. The linear approximation of the BoF model introduced has the added benefit that it is much faster than its non-linear counterpart, which is also a desirable feature in tracking tasks.

To track a target $\mathcal{T}$, first a target model $\mathbf{h}^{\mathcal{T}}$ is extracted as described. Then, given a new image frame, a candidate region $\mathcal{C}$ is extracted from an area around the previous target location, and modeled similarly as a candidate model $\mathbf{H}^{\mathcal{C}}$. For the candidate model, the same pooling grid as the one used for the extraction of the target model is used, e.g., $H \times W$ in the case of one global histogram for the target, or $n \times n$ in case of multiple histograms for different spatial locations of the target. The candidate region is chosen to be larger than the region corresponding to the target, thus producing a volume of histograms corresponding to various spatial locations.

Finally, the target histograms are compared against the candidate histograms using the χ^2 distance. For the case of one target histogram, the distances between the target histogram $\mathbf{h}^{\mathcal{T}} \in \mathbf{R}^M$ and each of the candidate histograms $\mathbf{H}^{\mathcal{C}} \in \mathbb{R}^{K \times L \times M}$ is given by:

$$d_{kl,m} = \sum_{m=1}^{M} \frac{(H_{kl,m}^{\mathcal{C}} - h_m^{\mathcal{T}})^2}{H_{kl,m}^{\mathcal{C}} + h_m^{\mathcal{T}}}. \tag{6}$$

for $k = 1, \ldots, K$ and $l = 1, \ldots, L$. The distances are then converted to similarities as follows:

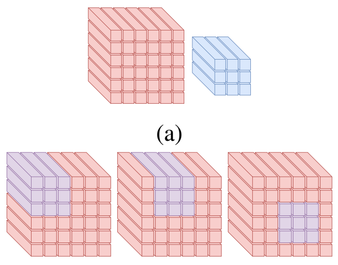
$$s_{kl,m} = \frac{\exp(-d_{kl,m})}{\sum_{n=1}^{M} \exp(-d_{kl,n})}. \tag{7}$$

In the case of the target model consisting multiple histograms, all histograms must be compared to the histograms corresponding to the candidate region in a convolutional fashion, and the similarity should be averaged over the target histograms. Figure 2 illustrates an example of this convolutional way of computing the similarities between multiple histograms.

After the similarities have been computed, tracking is simply a matter of finding the position at which the maximum similarity is achieved and projecting it back to the original image. To capture variations in scale, a scale pyramid is constructed from the candidate region and forward passed through the network as a single batch for efficiency. Variations in scale and large displacements are discouraged, using a penalty value and linearly combining the output with a cosine window respectively.

4. EXPERIMENTAL STUDY

Following [8], we use an AlexNet-like base architecture for our network, keeping only three convolutional layers and adding an LBoF [12, 13] layer at the end. The total stride of the network is 8. To extract target histograms, we crop the



(a)

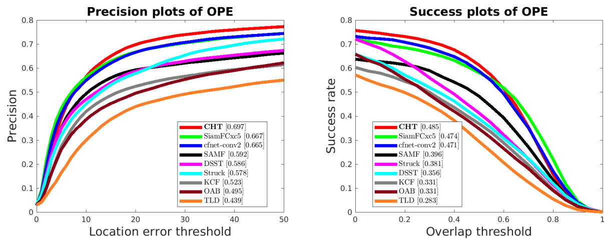
(b)

Fig. 2: (a) Candidate and target histograms (left and right respectively), extracted from convolutional feature vectors. (b) Different steps of comparisons between the target and candidate histograms. Purple blocks denote the position of the target histograms, overlayed on top of the candidate histograms to extract a single similarity value by averaging the similarities between the histograms of each overlapping grid cell.

region around the target including some context and resize the crop to be of fixed size 127×127 while maintaining the aspect ratio of the target. For the candidate histograms, we similarly resize the candidate region and crop an area of fixed size 255×255 around the previous location of the target. We train the network on the ILSVRC2015 VID dataset [14], learning the codewords of the LBoF layer from scratch. Following [9], we use the same validation set to stop the training process.

For evaluation, we use the OTB50, OTB2013, OTB100 [15, 16], and UAV123 datasets [17]. We compare our method against provided results from the SAMF [18], Struck [19], DSST [20], OAB [21], TLD [22], and KCF [23] trackers. We also compare our method against the more recently published SiamFC and CFNet [8, 9] and DCFNet [10] trackers when possible. The results for the One Pass Evaluation (OPE) on the UAV123 dataset are summarized in Figure 3. The proposed Convolutional Histogram Tracker (CHT) tracker achieves a precision score of 69.7% at pixel threshold 20, and a success AUC score of 48.5%, surpassing the compared trackers. By inspecting the attributes of the dataset, we observe that CHT surpasses all other trackers by a large margin on videos with Viewpoint Change and Aspect Ratio Change attributes, a feature that is inline with our hypotheses that histograms can more adeptly handle viewpoint changes.

The results for the OTB50, OTB2013 and OTB100 results are presented in Table 1. CHT achieves the best precision (71.7%) on the OTB50 dataset as well as challenging performance on the OTB2013 and OTB100 datasets. On the OTB datasets, our tracker achieves an average of about 120 frames per second using an NVIDIA GTX1080 Ti graphics card and an Intel i7-6900K CPU @ 3.20GHz CPU. On the same machine, its most competitive contender, SiamFC, runs at about 60 FPS. On an NVIDIA Jetson TX2 embedded module, our tracker can run at about 25 FPS.



(a) (b)

Fig. 3: (a) Precision curves at various thresholds for the UAV123 dataset. (b) Success curves at various overlap thresholds for the UAV123 dataset.

5. CONCLUSIONS

We have presented a novel tracker based on convolutional neural networks and a fully learnable Bag-of-Features module for histogram extraction. Based on the success of the Mean Shift algorithm, which uses color histograms, we hypothesize that using histograms of convolutional features instead of convolutional features can benefit the tracking task. Extensive experiments on multiple object detection benchmarks validate our assumptions. Finally, our tracker can run at realtime speeds even on embedded systems.

6. ACKNOWLEDGMENTS

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

7. REFERENCES

[1] Danai Triantafyllidou, Paraskevi Nousi, and Anastasios Tefas, "Fast deep convolutional face detection in the

Tracker	OTB50		OTB2013		OTB100	
	Precision	Success	Precision	Success	Precision	Success
CHT	**71.7**	51.6	*79.5*	58.9	76.1	*57.1*
SiamFC	*69.2*	51.6	**80.9**	*60.7*	**77.1**	**58.2**
CFNet	70.2	**53.0**	80.7	61.1	74.8	56.8
DCFNet	68.3	*50.9*	79.5	**62.2**	*75.1*	58.0
SAMF	63.2	45.6	77.0	56.6	74.3	53.5
Struck	50.2	36.6	64.1	47.4	58.4	42.9
DSST	62.5	41.1	73.7	50.3	69.3	47.0
KCF	61.1	40.3	74.0	51.4	69.5	47.7
OAB	37.6	27.3	50.7	37.0	49.0	36.6
TLD	42.9	32.2	55.3	40.7	54.6	40.6

Table 1: Precision and success scores for the OTB datasets.

wild exploiting hard sample mining," *Big data research*, vol. 11, pp. 65–76, 2018.

[2] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[3] Tomas Vojir, Jana Noskova, and Jiri Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[4] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. IEEE, 2000, vol. 2, pp. 142–149.

[5] Han B. Nam, H., "Learning multi-domain convolutional neural networks for visual tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg, "Eco: Efficient convolution operators for tracking," in *CVPR*, 2017.

[7] Silvio Savarese David Held, Sebastian Thrun, "Learning to track at 100 fps with deep regression networks," in *Conference Computer Vision (ECCV)*, 2016.

[8] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[9] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr, "End-to-end representation learning for correlation filter based tracking," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5000–5008.

[10] Qiang Wang, Jin Gao, Junliang Xing, Mengdan Zhang, and Weiming Hu, "Dcfnet: Discriminant correlation filters network for visual tracking," *arXiv preprint arXiv:1704.04057*, 2017.

[11] Olga Zoidi, Anastasios Tefas, and Ioannis Pitas, "Visual object tracking based on local steering kernels and color histograms.," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 23, no. 5, pp. 870–882, 2013.

[12] Nikolaos Passalis and Anastasios Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5755–5763.

[13] N. Passalis and A. Tefas, "Training lightweight deep convolutional neural networks using bag-of-features pooling," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2018.

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[15] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[16] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[17] Matthias Mueller, Neil Smith, and Bernard Ghanem, "A benchmark and simulator for uav tracking," in *European conference on computer vision*. Springer, 2016, pp. 445–461.

[18] Jianke Zhu Yang Li, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conference on Computer Vision, Workshop VOT2014 (ECCVW)*, 2014.

[19] Philip H. S. Torr Sam Hare, Amir Saffari, "Struck: Structured output tracking with kernels," in *International Conference on Computer Vision (ICCV)*, 2011.

[20] P. Martins J. Henriques, R. Caseiro and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *European Conference on Computer Vision (ECCV)*, 2012.

[21] Grabner M. Bischof H. Grabner, H., "Real-time tracking via on-line boosting," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2006.

[22] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas, "Face-tld: Tracking-learning-detection applied to faces," pp. 3789–3792, 2010.

[23] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.