ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

Continuous convolutional object tracking

Peer Springstübe, Stefan Heinrich, and Stefan Wermter *

Universität Hamburg - Dept. Informatics - Knowledge Technology Group
Vogt-Koelln-Str. 30 - 22527 Hamburg - Germany
http://www.informatik.uni-hamburg.de/WTM/

Abstract. Tracking arbitrary objects is a challenging task in visual computing. A central problem is the need to adapt to the changing appearance of an object, particularly under strong transformation and occlusion. We propose a tracking framework that utilises the strengths of Convolutional Neural Networks (*CNNs*) to create a robust and adaptive model of the object from training data produced during tracking. An incremental update mechanism provides increased performance and reduces training during tracking, allowing its real-time use.

1 Introduction

The process of tracking the changing position of an object is a very fundamental problem in the fields of computer vision as well as artificial intelligence and has been studied for a long time [1, 2, 3]. It is an important task in the area of artificial intelligence and robotics, as tracking objects visually is often a prerequisite for complex tasks. For example, a developmental robot can learn the affordance of a manipulated object by correctly keeping track of this object's change, while for an autonomous car keeping track of other vehicles and pedestrians in the streets can help planning a route and avoid collisions [4]. However, the visual tracking of arbitrary objects in a video stream is still a difficult task because the objects' appearance may change over time [5].

To overcome these challenges, an object tracker needs a mechanism for identifying and extracting robust features from a video stream. It also needs a flexible method for learning a model of the object's representation using a scarce supply of training data and adapting it dynamically over time. Previous attempts to provide such functionality include integrating CNNs for feature extraction and learning the object's visual representation. For instance, the Fully Convolutional Network based Tracker (*FCNT*) [6] uses the convolutional part of VGG16 [7] to extract visual features from a video stream and trains another small CNN on those features. Both the final and a penultimate convolutional layer were used as outputs since their features contribute different discriminating abilities. To improve the quality of the object representation, other algorithms include using pre-trained R-CNNs for feature extraction [8], training features in CNNs on the fly [3], or creating negative training samples by taking false locations around the correct position [9]. Nevertheless, they suffer from weak update selection schemes, making them prone to using poor samples and at the same time particularly complex, preventing any real-time application.

*The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169).

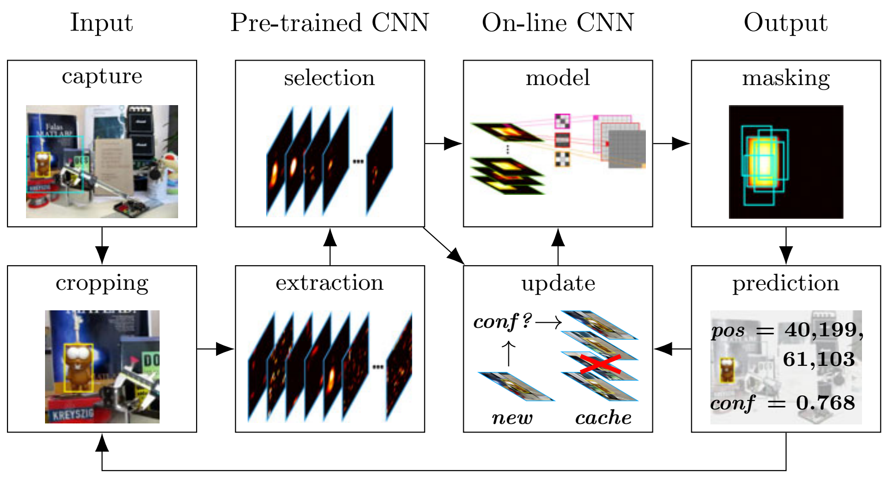


Fig. 1: Overview over the HIOB tracking framework.

In this paper, we propose *HIOB*, a HIerarchical and modular OBject tracking framework. HIOB provides CNN-based tracking with a robust update strategy and includes computationally effective training updates conditioned by the tracking confidence. Through this, HIOB provides robust tracking, particularly under occlusion and distortion conditions, and supports real-time applications.

2 Approach

Our proposed framework extends the FCNT introduced by Wang et al. [6]. It combines the feature extraction capabilities of a pre-trained CNN (*pCNN*) with the flexibility of an on-line CNN (*oCNN*), see Fig. 1 for an overview.

2.1 Initialising the Tracking

The tracking is initialised with a bounding box around the given position of the object in the first frame F^0. Next, the captured image is cropped to the region of interest (*ROI*) and scaled to a fixed size of 368×368 for input into the pCNN, which consists of the convolutional part of the VGG16 [7]. In particular, the last convolutional layer *conv5_3* and an earlier layer *conv4_3* are used for feature extraction and selection, since they have been shown to describe different visual aspects well [6]. A target heat map regression analysis is used to find the most impactful features for locating the object and to reduce the number of features per output layer from 512 to 384. These are used as input for the oCNN, which consists of two convolutional layers, where the first layer includes 32 filters with a kernel size of 9×9 and connects with a concatenated ReLU to the second layer, which comprises a single 9×9 filter. The oCNN is trained to produce the prediction mask in the form of a 2D-Gaussian, cropped to the object's bounding box on a 46×46 array.

2.2 Tracking

During tracking, for each following frame $F^t, t > 0$ the captured image is cropped around the last known position of the object, scaled, and fed as input to the pCNN. The resulting 2×384 features are used as input to the oCNN to produce a 46×46 mask for predicting the most probable areas of the object's position. This mask is normalised to values within $[0, 1]$, and integrated into a mask M^t representing the full sized captured image. On this prediction mask, 1 000 candidates for a new bounding box are created by altering the last predicted bounding box randomly, according to a Gaussian distribution. For each candidate X_n^t a confidence value $conf_n^t$ is calculated, with $A(X_n^t)$ being the size of X_n^t in pixels and $M^t(j)$ the probability predicted by the oCNN for pixel j:

$$conf_n^t = \frac{\sum_{j \in X_n^t} M^t(j)}{A(X_n^t)} \tag{1}$$

The candidate with the highest confidence is used as the predicted position X^t and its confidence is used as that prediction's confidence $conf^t$. If no candidate is rated above the threshold $min\_conf = 0.1$, the previous position is kept $(X^t = X^{t-1})$ and the confidence set to $conf^t = 0$.

Subsequent to every prediction, based on an update strategy, the sample is considered for updating the model. In this case, the sample is appended to a FIFO cache, storing the last 10 selected samples, and a single update iteration of the model is executed in its current configuration. Independent of the course of the tracking, the sample obtained from the initial frame is always kept in the cache because it is the only reference guaranteed to be of good quality.

2.3 Update Strategies

Deciding which samples are used for updating the model is critical for the success of the tracking. The algorithm must be able to adapt to changes in appearance to create a solid model, but at the same time must avoid samples of poor quality as they can corrupt the model. An update can be triggered by a prediction with a low confidence $conf^t$, which indicates a change in appearance. Updates should also get enforced after a certain number of frames δ_t without changing the model, in order to ensure that it is kept up to date. Utilising a lower bound for the confidence of samples prevents poor quality data to distort the model. In this work we explore four update strategies in detail:

- *None* – no updates executed to the model, serves as a baseline.
- *Full* – update on every frame, to explore the opposite extreme to *None*.
- *LCC* – update in case of low confidence $conf^t \leq 0.4$ combined with enforcing updates for $\delta_t = 20$, based on [6].
- *HGC* – update in cases of high gain, thus on low but not too low confidence $0.2 \leq conf^t \leq 0.4$, avoiding poor samples, enforced for $\delta_t = 20$.

Table 1: Precision (left) and success (right) on individual attributes.

Attrib.	None	Full	LCC	HGC	None	Full	LCC	HGC
all	0.653	0.469	0.796	0.840	0.489	0.356	0.541	0.566
BC	0.579	0.477	0.750	0.790	0.424	0.363	0.503	0.519
DEF	0.561	0.452	0.646	0.733	0.441	0.306	0.456	0.506
LR	0.719	0.575	0.948	0.936	0.365	0.183	0.432	0.423
MB	0.599	0.471	0.720	0.847	0.549	0.401	0.595	0.681
OCC	0.599	0.410	0.724	0.800	0.464	0.328	0.500	0.553
OV	0.608	0.344	0.773	0.814	0.505	0.326	0.613	0.642
SV	0.624	0.378	0.772	0.845	0.448	0.292	0.490	0.534

BC: Background Clutter, DEF: Deformation, LR: Low Resolution, MB: Motion Blur, OCC: Occlusion, OV: Out of View, SV: Scale Variation

3 Results and Evaluation

In order to evaluate our framework, we tested its performance on challenging data sets and live benchmarks and conducted in-depth case studies. In particular, we used the extension of the established Online Object Tracking Benchmark (*OOTB*) by Wu et al., which includes 100 tracking sequences with up to 3 872 frames and metrics for evaluation [10]. Furthermore, we participated in the Princeton Tracking Benchmark (*PTB*), which includes data and tests that are not publically available for framework or model optimisation [5].

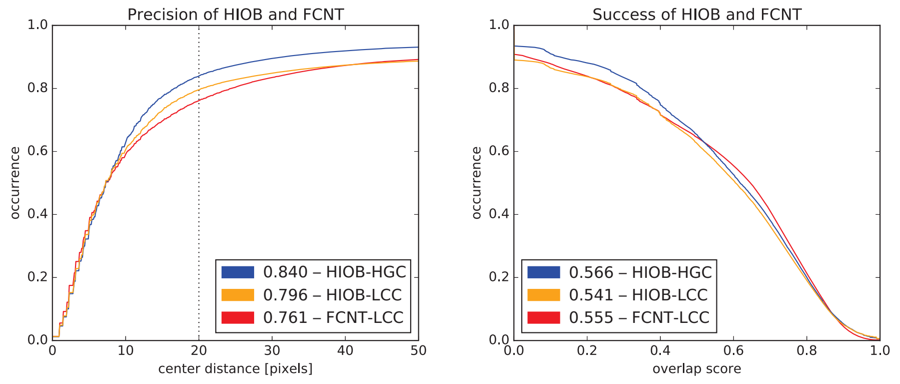
3.1 Performance

The OOTB has been used to evaluate the four update strategies. Tab. 1 provides the results on individual challenges included in the benchmark. Strategy *None* illustrates how HIOB performs without adaptation of the model during tracking. The poor performance by the *Full* strategy shows that more updates are not necessarily an improvement. *LCC* produces similar results in both frameworks as plotted in Fig. 2. The tracking experience in HIOB is much smoother compared to the FCNT because HIOB uses a single training step on updates when the FCNT reinitialises its network and executes 50 training steps for every update.

An analysis of failed trackings shows that the model is often disrupted by poor quality samples in the training data. These errors are amplified by the generation of additional erroneous predictions. *HGC* avoids this corruption by discarding samples of very low confidence. The result is a higher tracking performance with even fewer model updates. A significant performance increase can be seen for tracking sequences that include an occlusion of the object or motion blur which are likely to produce erroneous training samples (compare Tab. 1).

On the PTB[1], the HIOB framework ranks third out of 15 recent frameworks on the RGB challenge (not using depth information). It achieved the best results for the "animal" (72.5%) and "rigid" (78.2%) target types and ranks overall

[1]See `http://tracking.cs.princeton.edu/eval.php` for all current results.



(a) Precision plot and rating. (b) Success plot and rating.

Fig. 2: Comparison of the FCNT and the HIOB frameworks on the conventional metric as described in [10], using 20 pixels as the threshold for "good". The LCC strategy was tested in the original FCNT implementation from [6], and replicated within HIOB; HGC presents our proposed high gain strategy.

among the top three in all categories. A reason for not competing with the top two for the “human” (53.1%) target type seems to be the fact that these frameworks were particularly optimised for shapes, such as of human poses, by using RGB-D information. The results also indicate that HIOB is particularly good in non-occlusion (84.5%) and still quite good in occlusion cases (52.9%) that appear to be a major difficulty for all frameworks.

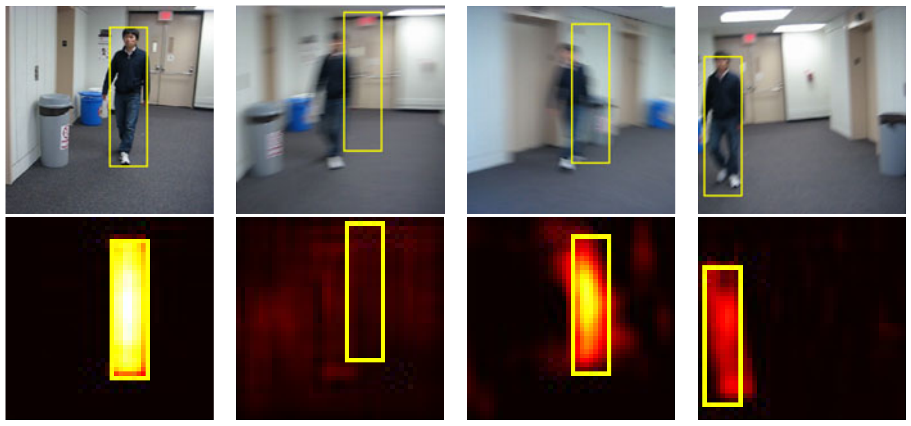
3.2 Case Study

The improved strategy was developed by analysing cases of failed trackings. Fig. 3 illustrates how poor quality samples are resulting in a model corruption. In Fig. 3b rapid camera movement and the caused motion blur produce a misplaced prediction. Because of the low confidence, an update is executed, training the model to predict a position behind the tracked person (Fig. 3c). With the HGC strategy, the updates are prevented until a less blurry image in a later frame, seen in Fig. 3d, produces a better training sample that gradually updates the model to recognise the person in a blurry image.

4 Discussion

A continuous model for object tracking, such as our proposed HIOB[2], can achieve a performance comparable to models that are constantly reinitialised when the model is utilising a smart update strategy. Our suggested strategy prevents disrupting the model by excluding poor quality data samples and simultaneously reduces the need for redundant updates. Overall, this leads to a significant

[2]The HIOB framework is available at https://github.com/kratenko/HIOB



(a) Initial frame shows a solid prediction. (b) Misplaced prediction caused by motion blur. Update executed on prediction with $conf = 0$. (c) Model was trained to predict the position next to the object. (d) The HGC strategy waited for a less blurry image to execute an update.

Fig. 3: Illustration how the LCC update strategy increasingly leads to predictions of low confidence, while HGC only utilises samples with a high gain.

improvement in the tracking performance and thus opens up applications that demand real-time and robust tracking. Future research includes further dynamic adaptations of the CNNs' complexities, based on the capabilities of the device and the context, e.g. in order to integrate HIOB in interactive humanoid robots.

References

[1] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, Aug 2011.

[3] H. Li, Y. Li, and F. Porikli. DeepTrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Trans. Image Process.*, 25(4):1834–1848, 2016.

[4] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[5] S. Song and J. Xiao. Tracking revisited using rgbd camera: Unified benchmark and baselines. In *Proc. ICCV 2013*, pages 233–240, 2013.

[6] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proc. ICCV 2015*, pages 3119–3127, Dec 2015.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *Proc. ICML 2015*, pages 597–606, 2015.

[9] X. Zhou, L. Xie, P. Zhang, and Y. Zhang. Online object tracking based on cnn with metropolis-hasting re-sampling. In *Proc. ICMR 2015*, pages 1163–1166, 2015.

[10] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proc. IEEE CVPR 2013*, pages 2411–2418, 2013.