Transferring Rich Feature Hierarchies for Robust Visual Tracking

Naiyan Wang†  Siyi Li†  Abhinav Gupta‡  Dit-Yan Yeung†
† Hong Kong University of Science and Technology  ‡ Carnegie Mellon University
winsty@gmail.com sliay@cse.ust.hk abhinavg@cs.cmu.edu dyyeung@cse.ust.hk

Abstract

Convolutional Neural Networks (CNNs) have demonstrated its great performance in various vision tasks, such as image classification [18] and object detection [8]. However, there are still some areas that are untouched, such as visual tracking. We believe that the biggest bottleneck of applying CNN for visual tracking is lack of training data. The power of CNN usually relies on huge (possible millions) training data, however in visual tracking we only have one labeled sample in the first frame. In this paper, we address this issue by transferring rich feature hierarchies from an offline pretrained CNN into online tracking. In online tracking, the CNN is also finetuned to adapt to the appearance of the tracked target specified in the first frame of the video. We evaluate our proposed tracker on the open benchmark [33] and a non-rigid object tracking dataset. Our tracker demonstrates substantial improvements over the other state-of-the-art trackers.

1. Introduction

The last two years have been exciting in the history of computer vision. This excitement has been mainly driven by biologically inspired convolutional neural networks (CNNs) which are trained in a supervised manner through backpropagation of errors. This theme has led to breakthrough performance on several high-level vision tasks such as image classification [18] and object detection [8]. However, some important research areas remain unexplored in this surge and one of them is visual tracking. Why is that? We believe the biggest driving force behind CNNs is their ability to learn with large-scale datasets (e.g., millions of images). In the case of visual tracking, however, we typically have only very few examples for training (in fact mostly we only have one single labeled example). This makes the direct applying of the large-scale CNN approach infeasible. In this paper, we present an approach to overcoming this problem and hence bringing the CNN framework to visual tracking. Our approach demonstrates state-of-the-art performance which outperforms the nearest baseline by more than 10% (see Fig. 1 for some qualitative tracking results).

Although visual tracking can be formulated under different settings, we focus on the one-pass model-free single object tracking setting in this paper. That is, we assume that the bounding box of one single object in the first frame is given but no other appearance model is available. Given a single (labeled) instance, the goal is to track the object in an online manner. Therefore, this setting involves adapting the tracker to appearance changes of the object based on the possibly noisy output of the tracker. Another way to formulate this problem would be as a self-taught one-shot learning problem in which the single example comes from the previous frame. Since learning a visual model from a single example is an ill-posed problem, a successful approach would require using some external data to learn an invariant representation of generic object features to exploit. Although there is some recent work [30, 32] that shares this spirit, the performance is severely hampered by the lack of sufficient training data and using an improper model. This is where CNN has a role to play. We propose to exploit the scalability of CNN to learn a good set of object features and then

![Tracking results on motocross1 and skiing sequences.](image)

Figure 1. Tracking results on motocross1 and skiing sequences.
transfer the features learned offline for the task of visual tracking.

To summarize, we propose a novel structured output CNN that transfers generic object features for online tracking. To prevent overfitting (and drifting at test time), the CNN is pre-trained to distinguish objects from non-objects instead of simply reconstructing the input on large-scale object-level annotation datasets [5]. During online tracking, the CNN is further tuned to adapt to the specific object being tracked. Furthermore, the output of the CNN is a pixel-wise map to indicate the probability that each pixel in the input image belongs to an object. The key advantages of the pixel-wise output are its induced structure loss and computational scalability. Since the output corresponds to a pixel probability map, first we could model the similarity between bounding boxes better, second we do not need to follow a proposal verification procedure in which hundreds of windows have to be tested at each frame. The pipeline of our tracking algorithm is depicted in Fig. 4. We evaluate our proposed algorithm on one open benchmark [33] and one challenging non-rigid object tracking dataset and show that our method significantly outperforms other state-of-the-art trackers.

2. Related Work

2.1. Deep Learning and CNNs

The root of deep learning can be dated back to research on multilayered neural networks in the late 1980s. The resurgence of interest in neural networks owes to a more recent work [15] which used pretraining to make training of deeper networks feasible. As for many vision tasks, CNNs seem to be a more suitable choice because, as inspired by their biological counterparts, they can capture better local and repetitive similarity in images by convolution and introduce translational invariance by pooling. The rapid development of powerful computing devices such as general-purpose graphics processing units (GPGPU) and the availability of large-scale labeled datasets such as [5] make the training of large-scale CNNs possible. It has been shown in [35] that a CNN can gradually learn low-level to high-level features through the transformation and enlargement of receptive fields in different layers. The features learned by a large-scale CNN has demonstrated superior performance in some high-level vision tasks, such as image classification [18] and object detection [8], as compared to conventional pipelines that use human-engineered features.

2.2. Visual Tracking

Many methods have been proposed for single object tracking over the past few decades. For a systematic review and comparison, we refer readers to a recent survey and benchmark [26, 33]. Most of the existing tracking methods belong to the general framework of Bayesian tracking [1]. It basically decomposes the problem into two parts which involve a motion model and an appearance model. Although some works have tried to go beyond this framework, e.g. [19, 20, 21, 24], most still focus on improving the appearance model as this aspect is crucial for enhanced performance.

Generally speaking, most trackers fall into two categories: generative trackers and discriminative trackers. Generative trackers usually assume a generative process of the tracked target and search for the most probable candidate as the tracking result. Some representative methods are based on principal component analysis [25], sparse coding [23], and dictionary learning [29]. On the other hand, discriminative trackers aim to separate the foreground from the background by a classifier. Many advanced machine learning algorithms have been used, including boosting variants [10, 11], multiple-instance learning [2], structured output SVM [12], and Gaussian process regression [7]. In general the two approaches to designing trackers are complementary. Discriminative trackers are usually more resistant to cluttered background since they explicitly sample image patches from the background as negative training examples. On the other hand, generative trackers are usually more accurate in normal situations. Besides, some methods exploit correlation filters for the target or context [4, 14, 37]. Their primary advantage is that only fast Fourier transform and several matrix operations are needed, making them very suitable for real-time applications. Some methods take the ensemble learning approach [3, 36, 31]. This approach is particularly effective when the constituent trackers involved in the ensemble are of great diversity. Also, some methods focus on long-term tracking, e.g. [17, 27].

As for applying deep learning to visual tracking, the first attempt was made recently by [30] in which a deep learning tracker (DLT) was proposed using a deep autoencoder network. While this work is interesting, there are two key issues that limit the performance of DLT. First, the pretraining of DLT may not be very suitable for tracking applications. The data used for pretraining are from the 80M Tiny Images dataset [28] with each image obtained by downsampling directly from a full image. Although some generic image features can be learned via learning to reconstruct input images, the target to track in a typical tracking task is a single object rather than an entire image. Features that are effective for tracking should be able to distinguish objects from non-objects (i.e. background), not just to reconstruct the entire image. Second, in each frame, DLT first generates candidates or proposals of the target based on the predictions of the previous frames, and then treats tracking as a classification problem. It ignores the structured nature of bounding boxes in that a bounding box or segmentation result corresponds to a region of an image, not just a simple label or real
number in a classification or regression problem. Some previous works \cite{12, 31} showed that taking the structured nature explicitly in the model could improve the performance significantly. Moreover, the number of proposals is usually in the order of several hundreds which may make bigger and more powerful deep learning models prohibitive. Some other approaches to the incorporation of deep learning in tracking include using an ensemble \cite{39} and maintaining a pool of CNNs \cite{22}. However, due to the lack of sufficient training data, these methods only show comparable or even inferior results compared to state-of-the-art trackers. In summary, for visual tracking applications, we believe the power of deep learning has not yet been fully released.

3. Our Model

In this section, we will present our structured output deep learning tracker (SO-DLT). We first present the CNN architecture and the offline pretraining process of SO-DLT. We then present details of its online tracking process.

3.1. Overview

Training of the tracker can be divided in two stages, offline pretraining and online fine-tuning and tracking. In the objectness pretraining stage, we train a CNN to learn generic features for distinguishing objects from non-objects. Thanks to the parametric form of the feature encoder of the CNN, we can fine-tune it during online tracking to make it adapt to the target being tracked instead of just fixing it after offline pretraining. More specifically, we run two CNNs in parallel during online tracking to account for possible mistakes made by model update. The two CNNs work collaboratively in determining the final result of each frame.

3.2. Objectness Pretraining

The architecture of the structured output CNN is shown in Fig. 2. It basically shares the same spirit as the CNN used in \cite{18}. It consists of seven convolutional layers and three fully connected layers. Between these two parts, a multiscale pooling scheme \cite{13} is introduced to emphasize locality. The parameter settings of the network are all shown in Fig. 2. The input to the CNN is of size $100 \times 100$ and the output is a $50 \times 50$ probability map. Each output pixel corresponds to a $2 \times 2$ region in the original input, with its value representing the probability that the corresponding input region belongs to an object. In order to train such a large CNN, a large dataset is essential to prevent overfitting. Since we are interested in object-level features, we use the ImageNet 2014 detection dataset\footnote{http://image-net.org/challenges/LSVRC/2014/} which contains 478,807 bounding boxes in the training set. For each annotated bounding box, we add random padding and scaling around it. We also randomly sample some negative examples when the overlap rates\footnote{The overlapping rate between two bounding boxes is defined as the area of union of two bounding boxes over the area of union of them.} of the positive examples are below a certain threshold. Note that it does not learn to distinguish different classes as in a classification or detection task, since we are only interested in learning to differentiate objects from non-objects in this stage. Consequently, we use an element-wise logistic regression model in each position of the $50 \times 50$ output map and define the loss function accordingly. The detailed parameters for training are described in Sec. 4.1. Fig. 3 shows some testing results of the proposed CNN on the held-out validation set provided by the ImageNet 2014 detection task. In most cases, the CNN can successfully determine whether the input image contains an object, and if yes it can accurately locate the object of interest.

![Testing of the pretrained objectness CNN on the ImageNet 2014 detection validation set.](image)

Figure 3. Testing of the pretrained objectness CNN on the ImageNet 2014 detection validation set. The first row shows two positive examples each of which contains an object. The objectness CNN can accurately detect the object positions and scales. The second row shows two negative examples. The objectness CNN does not fire on them. This makes our SO-DLT robust to occlusion and cluttered background during online tracking.

3.3. Online Tracking

In the previous section, we discussed how we can exploit ImageNet data to pretrain SO-DLT and learn generic object features. However, directly using them for online tracking is not appropriate because the data bias in tracking is different from that in ImageNet dataset. Therefore, it is essential to fine-tune the pretrained CNN using the annotations in the first frames of the videos collected during online tracking.

The basic pipeline for online tracking is outlined as follows. We maintain two CNNs which use different model update strategies. After fine-tuning with the first frame, we crop some samples from each new frame based on the estimation of the previous frame. By making a simple forward pass through the CNN, we can obtain the probability map of these samples. The probability value of each pixel in-
dicates the confidence that the pixel belongs to the target, and the prediction confidence of a whole region is just the average confidence over all pixels in the region. The final estimation is then determined by thresholding the map and searching for a continuous region with all its pixels detected as belonging to the object. The two CNNs are updated if necessary. We illustrate the pipeline of the tracking algorithm in Fig. 4. In what follows, we will elaborate different steps of the pipeline separately.

3.3.1 Differentially-paced Fine-tuning

The issue of model update in visual tracking often faces a dilemma. If the tracker does not update frequently enough, it may not adapt well to appearance changes. However, if it updates too frequently, inaccurate results may impair its performance and lead to the drifting problem.

We tackle this dilemma by using two CNNs in the online tracking process. The basic idea is to make one CNN (CNN_S) account for short-term appearance while the other one (CNN_L) for long-term appearance. First, both CNNs are fine-tuned in the first frame of a video. Afterwards, CNN_L is tuned conservatively while CNN_S is tuned aggressively. By working collaboratively, CNN_S adapts to dramatic appearance changes while CNN_L is resistant to potential mistakes. The final estimation is then determined by the more confident one. Consequently, the final integrated results are more robust to drifting caused by occlusion and a cluttered background. Although there exist more advanced model update methods, e.g. [36], we find that our simple scheme works quite well in practice.

We now provide more details about the update strategies. We first observe that if a model is updated immediately as soon as the confidence of prediction falls below a threshold, the model will be easily influenced by noisy results. On the other hand, we find that the quality of negative examples is generally quite stable. As a result, if there exists a negative example that the sum of the output map is above a threshold $\tau_1$, which means our tracker could possibly drift on it in the following frames, then we update CNN_S. In contrary, the CNN_L will only be updated if the average confidence of the predicted bounding box of CNN_L in one frame is greater than a threshold $\tau_2$ in addition to the condition applied to CNN_S. Doing so will reduce the risk of incorrect update when the true target has already drifted into the background.

3.3.2 Sample Collection

For each update, we need to collect both positive and negative examples. Our sampling scheme is illustrated in Fig. 5. For positive samples, we sample them in four scales based on the estimation in previous frame. Random translation is also introduced to eliminate the learning bias to the center location. While for the negative ones, we crop 8 non-overlapping bounding boxes around the target in different directions in two scales. The output of the positive samples
are also shown in Fig 5.

Figure 5. Sampling scheme of the proposed tracker. On the left, the red bounding box denotes the target to track while the eight blue ones around it are negative examples. On the right, we show positive examples fed to the CNN in the upper part. They are padded with different scales and random translations. The lower part shows the corresponding output from the CNN after applying fine-tuning to this frame.

3.3.3 State Determination

Our method outputs a probability map to indicate the status of the target. This is sufficient for most real applications. However, in order to make a fair comparison with other trackers, we need to decide the best bounding box based on the probability map.

For each test sample, we set a different threshold for the corresponding probability map and find a bounding box that contains all the values in the map above the threshold. Next, the bounding box location under the current scale is estimated by taking an average over the different thresholds. After the center is determined, we need to search again in the corresponding region to find a proper scale. The scale is aimed at fitting the exact target region perfectly. We first subtract the probability map by a threshold $\epsilon$. However, directly finding a bounding box that maximizes the sum of confidence within it is not a proper choice, since it is sensitive to the value of $\epsilon$, and many candidates with different scales may have similar sum of confidence. Consequently, for each candidate region, we propose to multiply the sum of confidence by the area of the candidate. Therefore, if two candidates have similar positive confidence values, we are encouraged to choose the larger one. On the other hand, for the candidates with negative confidence values, we prefer the smaller one. This weighting scheme ensures that our tracker can adapt well to scale changes. We also iterate over several $\epsilon$ values and average their results for robust estimation. Note that this approach shares the same spirit as that in [34] which is also based on a confidence map.

Nevertheless, determining a proper search range is a non-trivial problem. Using too small search regions makes it easy to lose track of a target in fast motion, but using too large search regions may include salient distractors in the background. For example, in Fig. 5, the output response gets weaker as the search region is enlarged, mainly due to the cluttered background and another person nearby. To address this issue, we propose a multi-scale search scheme for determining the proper bounding box. We start with the smallest scale. If the average confidence within the resulting bounding box is below a threshold (i.e., the target may not be in this scale) or part of the bounding box is out of range (i.e., the true target may be truncated in this scale), then we proceed to the next scale. If we cannot find the object in all scales, we report that the target is missing.

4. Experiments

In this section, we empirically compare the proposed tracker SO-DLT with other state-of-the-art trackers. Testing a tracker on a small number of sequences might cause biased evaluation due to sequence selection and parameter tuning. Recent work [33] introduced a unified tracking benchmark to reduce the bias. We evaluate the proposed tracker on this benchmark by strictly following their protocol and fixing all the parameters. We will make our codes publicly available if the paper is accepted.

4.1. Implementation Details

The part related to CNN is implemented using Caffe toolbox$^3$, while the wrapper for online tracking is implemented in MATLAB. All of the experiments are evaluated on a desktop PC with a 3.40GHz CPU and a K40 GPU. The speed of our unoptimized codes is about 4 to 5 frames per second.

For the pre-training of CNN, we start with a learning rate of $1e^{-7}$ with momentum 0.9 and decrease it every 5 epochs. We train about 15 epochs in total. Note that the learning rate is much smaller than the typical setting due to the different loss function we use. To alleviate overfitting, a weight decay of $5e^{-4}$ is used for each layer, and the first fully connected layer is regularized with dropout ratio 0.5. In fine-tuning, we use a larger learning rate of $2e^{-7}$ with a smaller momentum 0.5. In the first frame, we tune twenty iterations for each CNN. For subsequent fine-tuning in the tracking process, we only finetune for one iteration.

The sum of confidence $\tau_1$ for negative samples is set as $\tau_1 = 100$. The updating threshold of $CN_{\tau}$ is set as $\tau_2 = 0.8$. And the normalized constant for searching proper scales $\epsilon$ is ranged from 0.55 to 0.6 with a step of 0.025.

4.2. CVPR2013 Visual Tracker Benchmark [33]

The CVPR2013 Visual Tracker Benchmark contains 50 fully annotated sequences and covers a variety of challenging scenarios in the online tracking literature over the past few years.

---

$^3$http://caffe.berkeleyvision.org
4.2.1 Evaluation Setting and Metrics

We use two performance measures analogous to the area under curve (AUC) measure for the receiver operation characteristic (ROC) curve. Specifically, for a given overlap threshold in [0, 1], a tracker is considered successful in a frame if its overlap rate exceeds the threshold. The success rate measures the percentage of success frames over an entire video. By varying the threshold gradually from 0 to 1, it gives a plot of the success rate against the overlap threshold for each tracker. A similar performance measure called precision plot is also defined for the central-pixel error. The difference is that the precision at threshold 20 is used for the performance score instead of the AUC score for success plots. They are more informative and can accurately reflect a tracker’s performance compared to some widely used metrics such as the success rate and the average center location error which only use only one threshold. All the results of 29 trackers on the benchmark can be found in [33]. For a more complete comparison, we also include a recent tracker TGPR [7]. As far as we know, TGPR is the best single tracker with code publicly available.
Table 1. Success rates at different thresholds based on the overlap rate metric for different tracking methods.

| Method | p@0.3 | p@0.5 | p@0.7 |
|--------|-------|-------|-------|
| SO-DLT | 0.8576 | 0.7798 | 0.4732 |
| TGPR   | 0.7690 | 0.6463 | 0.3767 |
| SCM    | 0.6807 | 0.6162 | 0.4396 |
| Struck | 0.6694 | 0.5593 | 0.3543 |
| TLD    | 0.6362 | 0.5210 | 0.2997 |
| DLT    | 0.5912 | 0.5066 | 0.3581 |
| ASLA   | 0.5730 | 0.5112 | 0.3884 |

Table 2. Success rates at different thresholds based on the central-pixel error metric for different tracking methods.

| Method | p@15 | p@25 | p@35 |
|--------|------|------|------|
| SO-DLT | 0.7721 | 0.8470 | 0.8772 |
| TGPR   | 0.7133 | 0.7907 | 0.8215 |
| Struck | 0.6047 | 0.6895 | 0.7257 |
| SCM    | 0.6171 | 0.6703 | 0.6980 |
| TLD    | 0.5520 | 0.6403 | 0.6848 |
| DLT    | 0.5400 | 0.6128 | 0.6541 |
| ASLA   | 0.5050 | 0.5554 | 0.5915 |

Table 3. Central pixel errors of various algorithms on the non-rigid object tracking dataset.

| Sequence    | SO-DLT | TGPR | PixelTracker |
|-------------|--------|------|--------------|
| cliff-dive1 | 14.36  | 32.77 | 23.12        |
| cliff-dive2 | 15.04  | 17.65 | 35.86        |
| motocross1  | 14.49  | 188.09 | 125.71       |
| motocross2  | 12.18  | 11.95 | 31.76        |
| skiing      | 5.85   | 282.00 | 7.22         |
| mountain-bike | 6.10  | 5.48  | 215.05       |
| volleyball  | 83.77  | 7.24  | 115.86       |
| average     | 21.69  | 77.88 | 79.26        |

4.2.2 Quantitative Results

Due to space limitations, we only show the overall performance of one pass error (OPE) for our proposed tracker and compare it with some of the state-of-the-art trackers (ranked within top 5) in each plot. These trackers include Struck [12], SCM [38], TLD [17], ASLA [16], DLT [30] and TGPR [7]. The success and precision plots are shown in Fig. 6. A tracker’s curve is computed by taking the average on all 50 test sequences. From Fig. 6, it can be seen that SO-DLT substantially outperforms other trackers by a significant gap in all evaluation settings. For example, in the success plot, our tracker SO-DLT outperforms the second best tracker TGPR by 13.8%; and in the precision plot, SO-DLT beats TGPR by 6.9%. When compared with other trackers ranking top in the benchmark, the improvement of SO-DLT is much more significant.

We also report the average success rates at several thresholds for different methods in Tab. 1 and Tab. 2. Our SO-DLT always ranks first, especially in the middle ranged threshold: For example, at the overlapping rate of 0.5 (which is always called “successful rate”), we outperform the nearest baseline by 20.6%. We believe the comparison is substantial enough to demonstrate the superiority of our SO-DLT.

For better analysis of the strength and weakness of each tracker, each of the 50 sequences is also annotated with attributes that indicate the challenging factor in it. Fig. 7 shows the AUC ranking scores of the leading trackers on different groups of sequences, where each group indicates a different attribute. From Fig. 7, we can see that SO-DLT substantially outperforms the other state-of-the-art trackers in almost all conditions. More precisely, our proposed tracker gets the highest score among all the 11 attributes. Besides, we can also observe that SO-DLT is especially good at handling some attribute groups such as “illumination variation (IV)”, “out-of-plane rotation (OPR)”, “in-plane rotation (IPR)”, “out-of-view (OV)” and “scale variation (SV)”. These sequences often suffer from large appearance changes. The observation demonstrates the great representation power of CNN in detecting the target object even in some extreme cases. Moreover, when model drifting happens, the proposed differentially-paced finetuned CNN can also help correct the drifting soon once the occlusion disappears.

Note that due to space limitation, we only include the above representative results and leave more details in the supplementary material.

4.3. Non-rigid Object Tracking Dataset

To gain a deeper insight of the proposed SO-DLT, we conduct an additional analysis on a challenging non-rigid object tracking dataset proposed in [9].

In this experiment, we only compare with the best performed tracker TGPR [7] in the previous benchmark, and a state-of-the-art non-rigid object tracker PixelTracker [6]. Since the target is highly deformable, comparing the overlapping rate defined by bounding box is meaningless. So we only use central pixel error as our evaluation metric. The results are shown in Tab. 3.

As we can see, though not designed to tracking non-rigid object on purpose, our SO-DLT can still outperform dedicated PixelTracker [6]. Compared to the universal tracker TGPR [7], the improvements are even more significant. Our tracker can always track the target to the end except the volleyball sequence. We show additional visual results in Fig. 8.
4.4. Discussion

Although our tracker demonstrates superior performance in the benchmarks, it is still far from perfect. Here we analyze the failure cases, and discuss possible improvements here:

1. The tracker is likely to drift when there exists distractor in background and the true target is occluded. Because of the powerful and invariant representation power of CNN, the distractors with similar appearance may be recognized as the target, thus cause incorrect tracking results. This is just why SO-DLT is a bit worse than TGPR in the “background cluttered” in the benchmark.

2. The tracker can possibly drift when the initial bounding box is not proper. This case often occurs when the shape of the target is irregular, so that a small portion of the entire object is specified as target.

To solve these challenging issues, we think there are three possible solutions:

1. In addition to directly transferring high level features in SO-DLT which introduces too much invariance for the cases contain distractors, we could build another tracker that only transfers low level features that capture the local invariance of appearance. These two trackers could work collaboratively to decide the best state.

2. More advanced model update methods could be tried. Model ensemble based methods [36, 31] have shown great potential recently. They can detect and correct the mistakes in the model update by undoing some incorrect model updates or reweighting each model adaptively.

3. To better handle irregular object shape, our model is readily to output a pixel-wise map of target in case that there is proper data for pretraining and initialization in the first frame.

However, these improvements are all beyond the scope of this paper, we would like to investigate them in our future work.

5. Conclusion

In this paper, we have exploited the effectiveness of transferring high-level feature hierarchies for visual tracking. To the best of our knowledge, we are the first to bring large-scale CNN to the area of visual tracking and show significant improvement over the state-of-the-art trackers. Instead of modeling tracking as a proposal classification problem, we presented a novel structured output CNN for visual tracking. Instead of reconstructing the input image, the CNN is first pretrained on the large-scale ImageNet [5] detection dataset to localize objects so as to alleviate the lack of data in visual tracking. Then this objectness CNN is transferred and fine-tuned during the online tracking process. Extensive experiments validate the superiority of our proposed SO-DLT.

References

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing, 50(2):174–188, 2002.

[2] B. Babenko, M. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(8):1619–1632, 2011.

[3] C. Bailer, A. Pagani, and D. Stricker. A superior tracking approach: Building a strong tracker through fusion. In ECCV, pages 170–185, 2014.

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 2544–2550. IEEE, 2010.
[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. ImageNet: A large-scale hierarchical image database. In CVPR, pages 248–255, 2009. 2, 8

[6] S. Duffner and C. Garcia. PixelTrack: a fast adaptive algorithm for tracking non-rigid objects. In ICCV, pages 2480–2487, 2013. 7

[7] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with Gaussian processes regression. In ECCV, pages 188–203, 2014. 2, 7

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524, 2013. 1, 2

[9] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. Computer Vision and Image Understanding, 117(10):1245–1256, 2013. 7

[10] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In BMVC, pages 47–56, 2006. 2

[11] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In ECCV, pages 234–247, 2008. 2

[12] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In ICCV, pages 263–270, 2011. 2, 3, 7

[13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, pages 346–361, 2014. 3

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. arXiv preprint arXiv:1404.7784, 2014. 2

[15] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006. 2

[16] X. Jia, H. Lu, and M. Yang. Visual tracking via adaptive structural local sparse appearance model. In CVPR, pages 1822–1829, 2012. 7

[17] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7):1400–1422, 2012. 2, 7

[18] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, pages 1106–1114, 2012. 1, 2, 3

[19] J. Kwon and K. Lee. Visual tracking decomposition. In CVPR, pages 1269–1276, 2010. 2

[20] J. Kwon and K. M. Lee. Minimum uncertainty gap for robust visual tracking. In CVPR, pages 2355–2362, 2013. 2

[21] J. Kwon and K. M. Lee. Interval tracker: Tracking by interval analysis. In CVPR, pages 3494–3501, 2014. 2

[22] H. Li, Y. Li, and F. Porikli. DeepTrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. 2014. 3

[23] X. Mei and H. Ling. Robust visual tracking using $l_1$ minimization. In ICCV, pages 1436–1443, 2009. 2

[24] D. W. Park, J. Kwon, and K. M. Lee. Robust visual tracking using autoregressive hidden markov model. In CVPR, pages 1964–1971, 2012. 2

[25] D. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. International Journal of Computer Vision, 77(1):125–141, 2008. 2

[26] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(7), 2014. 2

[27] J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In CVPR, pages 2379–2386, 2013. 2

[28] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(11):1985–1970, 2008. 2

[29] N. Wang, J. Wang, and D.-Y. Yeung. Online robust non-negative dictionary learning for visual tracking. In ICCV, pages 657–664, 2013. 2

[30] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In NIPS, pages 809–817, 2013. 1, 2, 7

[31] N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. pages 1107–1115, 2014. 2, 3, 8

[32] Q. Wang, F. Chen, J. Yang, W. Xu, and M. Yang. Transferring visual prior for online object tracking. IEEE Transactions on Image Processing, 21(7):3296–3305, 2012. 1

[33] Y. Wu, J. Lim, and M. Yang. Online object tracking: A benchmark. In CVPR, 2013. 1, 2, 5, 7

[34] F. Yang, H. Lu, and M.-H. Yang. Robust superpixel tracking. IEEE Transactions on Image Processing, 23(4):1639–1651, 2014. 5

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, pages 818–833, 2014. 2

[36] J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In ECCV, pages 188–203, 2014. 2, 4, 8

[37] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang. Fast visual tracking via dense spatio-temporal context learning. In ECCV, pages 127–141, 2014. 2

[38] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In CVPR, pages 1838–1845, 2012. 7

[39] X. Zhou, L. Xie, P. Zhang, and Y. Zhang. An ensemble of deep neural networks for object tracking. In ICIP, 2014. 3