Re^3: *Re*al-Time *Re*current *Re*gression Networks for Object Tracking

Daniel Gordon[1] Ali Farhadi[1,2] Dieter Fox[1]

Paul G. Allen School of Computer Science and Engineering, University of Washington[1]
Allen Institute for Artificial Intelligence[2]
{danielgordon, ali, fox}@cs.washington.edu

Abstract

Robust object tracking requires knowledge and understanding of the object being tracked: its appearance, its motion, and how it changes over time. A tracker must be able to modify its underlying model and adapt to new observations. We present Re^3, a real-time deep object tracker capable of incorporating long-term temporal information into its model. In line with other recent deep learning techniques, we do not train an online tracker. Instead, we use a recurrent neural network to represent the appearance and motion of the object. We train the network offline to learn how an object's appearance and motion may change, letting it track with a single forward pass at test time. This lightweight model is capable of tracking objects at 150 FPS, while attaining competitive results on challenging benchmarks. We also show that our method handles temporary occlusion better than other comparable trackers using experiments that directly measure performance on sequences with occlusion.[1]

1. Introduction

Object tracking plays an important role in many real-world computer vision applications. Tracking algorithms must be able to find a variety of objects while simultaneously handling visual challenges such as occlusion, lighting change, deformation, and change in scale and aspect ratio.

The problem of generic 2D object tracking can be concisely specified as as: given a bounding box around an object at the start of a video, track the object for the remainder of the video. Current systems predominantly rely on learning a tracker online. A popular paradigm for tracking algorithms is tracking-by-detection: training an object-specific detector or classifier, and updating the classifier with the object's new appearance at every frame. A disadvantage of this technique is that updating the tracker takes time and computational resources. It also makes the assumption that there is no valuable information that the detector could learn about the object before the first frame. No matter how capable an online detector is, it cannot recognize parts of the object it has never seen, nor can it predict the expected motion of the object.

A robust object tracker needs to model low level object characteristics such as color and high level characteristics such as category. It must also understand and encode the likely motion of an object in order to keep track during occlusions. Finally, it must understand how the object changes appearance over time.

We propose a method using a deep recurrent neural network that addresses these requirements. We show that given enough examples, a deep neural network can learn these underlying correlations before the tracking task even begins. By incorporating information from large collections of images and videos, our network learns to produce representations that better capture the important features of the tracked object. The goal of this process is to teach the network how objects are likely to change over time so that these transformations can be embedded directly into the network. This shifts the computational burden offline so that our tracker can be extremely fast at test time.

Deep convolutional neural networks have been shown to advance the state-of-the-art for multiple related tasks such as object recognition [26], object detection [56], and image pixel labeling [71]. Generic object tracking represents a new challenge for deep methods. Most deep learning algorithms rely on having millions of examples to function, learning invariance to high-level concepts; object recognition, object detection, and image captioning algorithms all expect the network to understand what a person looks like, but not necessarily to differentiate between two people. Trackers, on the other hand, are often given only a single initial example and must specialize in finding that specific object. Because of the difficulty of adapting traditional deep methods to tracking, deep learning has only recently started to be used in tracking algorithms. In 2015, MDNet [52], a deep method won the The Visual Object Tracking challenge (VOT) [41] for the first time. However this system was slow, operating at less than one frame per second,

[1] https://youtu.be/PC0txGaYz2I

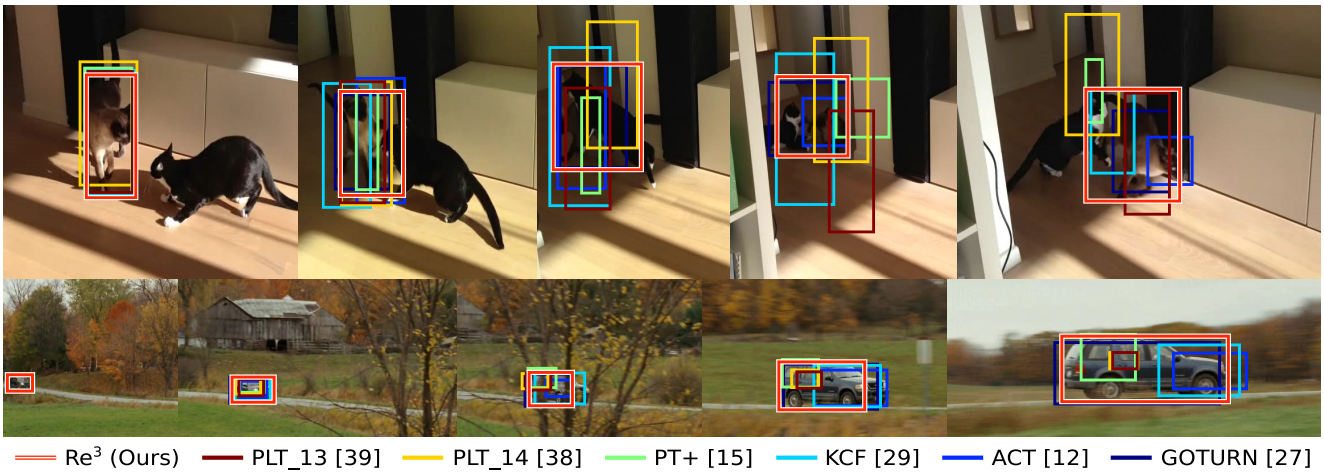

Figure 1. Various real-time tracking algorithms run on VOT 2014 sequences [38]. Our method is robust to occlusions, small objects, changing aspect ratios, and varying object viewpoints. Trackers used: [39, 38, 15, 29, 12, 27]

and was not an end-to-end trainable system.

We propose the *Re*al-time, *Re*current, *Re*gression-based tracker, or Re^3: a simple, fast, and accurate network for generic object tracking. Re^3 has various benefits over traditional tracking-by-detection approaches and comparable deep methods. Unlike MDNet [52], our method is fully end-to-end trainable, meaning we teach the network to track directly rather than relying on learning an additional classifier at test time. Though we do not train a classifier online, our method can still update its underlying representation of the object via the recurrent parameters, a feature which other recent deep, real-time trackers lack [4, 27]. This lets us model temporal dependencies between sequential images and reason about occlusions.

Our network differs from prior work in that we use a recurrent structure known as an LSTM (Long Short-Term Memory) rather than a feed-forward network. By using an LSTM, we allow the network to learn to remember, forget, and ignore information about the appearance and motion of the object. The recurrent parameter update is performed during the forward pass, requiring no extra computational burden, allowing our tracker to operate at 150 frames per second. Furthermore, LSTMs allow us to handle occlusions directly. By training our network on many clips with occlusions, we encourage the LSTM to use its gates to ignore appearance changes caused by the occluders. We show that our model can learn robustness to occlusion, outperforming other trackers in these cases. Finally, by training on thousands of example videos and millions of images, our tracker learns what natural objects look like, and how their appearance and motion tend to change over time. This directly contrasts the many tracking-by-detection methods which start each new track as a blank slate. Our results show that recurrent networks are well suited for object tracking, as they can be fast, accurate, and robust to occlusions. Re^3 achieves competitive results on multiple tracking benchmarks, showing especially good performance during occlusions, all while running at 150 frames per second.

2. Related Work

In computer vision, generic object tracking has been studied in great depth. The VOT reports [38, 41, 42] present a succinct overview of many state-of-the-art trackers. Those most related to ours can be categorized into three sub-groups: online-trained, offline-trained, and hybrid trackers.

Online-trained trackers: The most prevalent type of trackers operate entirely online, continually learning features of the object of interest as new frames arrive. This includes keypoint-based and part-based trackers [15, 17, 45, 53], correlation based methods [29, 46], and direct classification methods [2, 16, 24]. These operate entirely online, often by rapidly training a classifier to differentiate between the object of interest, the background, and possible occluders [9, 24, 34]. Discriminative Scale Space Tracker (DSST) [9], the winner of the VOT 2014 challenge [38], uses this approach. DSST learns discriminative correlation filters for different scale and translation amounts. Because online trackers must train on frames as they arrive, they tend to directly trade off speed with model complexity.

Offline-trained trackers: The success of deep learning is often attributed in part to its ability to utilize massive amounts of training data better than other machine learning methods. Offline trackers such as [4] and [27] employ this technique to great success. Because they are trained entirely offline, the networks are fast to evaluate at test time, allowing both methods to operate at faster than real-time speeds. However, this underscores a large problem with offline trackers: they do not adapt to what they are seeing. Instead of incorporating information from an entire track, they

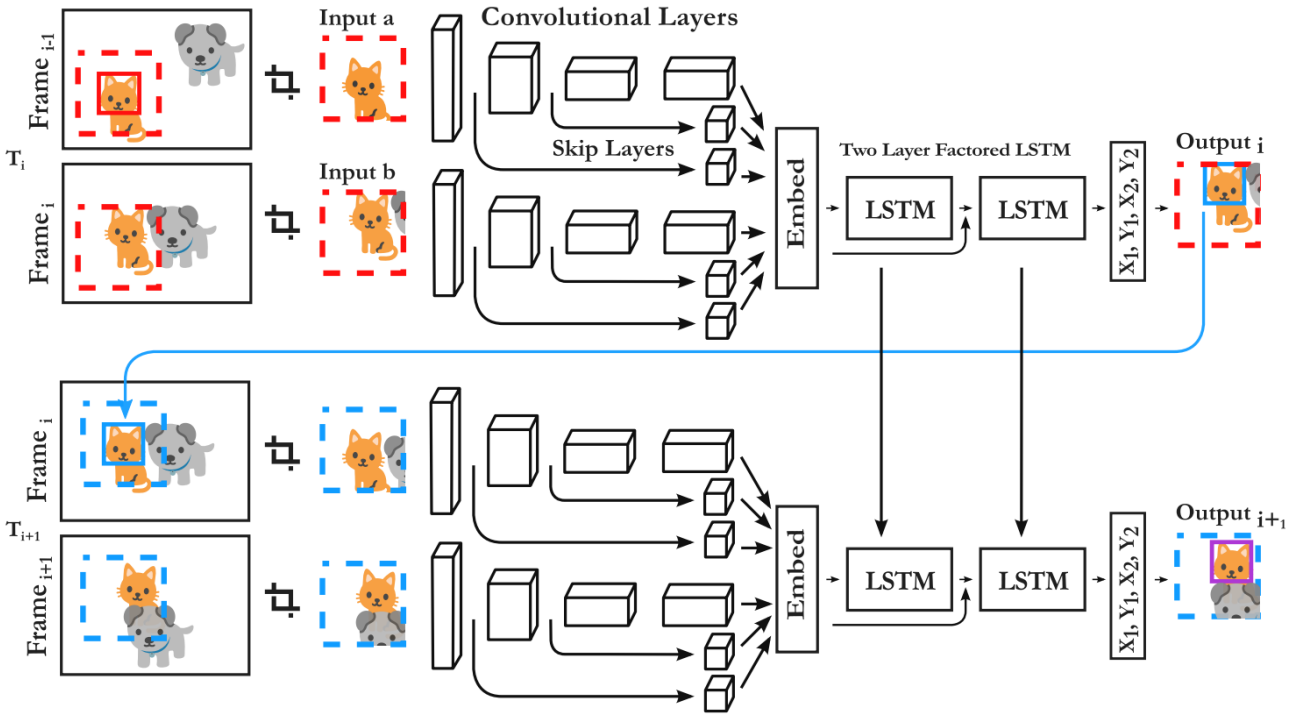

Figure 2. Network Structure: Image pair crops are fed in at each timestep. Both crops are centered around the object's location in the previous frame, and padded to two times the width and height of the object. The output from the convolutional layers feeds into a fully-connected embedding layer which is then passed to a two layer factored LSTM. The network outputs a prediction for the top left and bottom right corners of the new bounding box. Dotted boxes of the same color represent crops that all have the same XY coordinates, though they may be in different images. In the example scene, the cat's face is the object of interest, and the dog is an occluder.

learn a similarity function between pairs of frames. Held *et al*. [27] use only a single frame history, meaning any amount of occlusion will confuse the tracker. Bertinetto *et al*. [4] rely solely on the initial frame for appearance information and try to detect the object in all subsequent frames, meaning large appearance changes, even if gradual, would be difficult to track. CUVideo [36] focuses on the detection and tracking problem in video by finding temporally coherent object detections, but they do not adapt the model using visual information from prior detections. Offline trackers' capabilities are limited because they cannot adapt to new information.

Hybrid trackers: Hybrid trackers attempt to solve the problems with online and offline trackers by taking the best from both approaches. MDNet, the winner of the VOT 2015 challenge, trained an image classification network offline, and then learned a per-object classifier online [52]. Similar approaches were taken by other top competitors [10, 66]. Still, the complexity of their online training techniques limited their methods to taking seconds to process each frame.

Our approach is a hybrid tracker, but prioritizes offline learning and limits online adaptation to recurrent state updates. Although we make this trade-off, our method is substantially different from purely offline trackers because we use information from previous frames to make future predictions. Other recurrent trackers such as [19] and [33] use attention-based recurrent neural networks. These techniques have only been shown to work on simple datasets such as tracking MNIST digits. To our knowledge, we are the first to demonstrate successful tracking in natural videos using recurrent neural networks.

3. Method

Our tracking pipeline, depicted in Figure 2, consists of convolutional layers to embed the object appearance, recurrent layers to remember appearance and motion, and a regression layer to output the location of the object. We train this network on real videos, using synthetic data to augment the training set. At test time, unlike MDNet [52], we do not update the network itself; we instead let the recurrent parameters represent the tracker state which can be updated with a single forward pass. In this way, the tracker learns to use new observations to update the hidden appearance and motion models, but no extra computational cost is spent on online training.

3.1. Object Appearance Embedding

The task of generic object tracking in video sequences starts with an initial bounding box around an object, with

the goal of keeping track of that object for the remainder of the video. For each frame of the video, the tracker must locate the object as well as update its internal state so it can continue tracking in future frames. A primary subtask in this framework is translating raw pixels into a higher-level feature vector representation. Many object trackers, like [17] and [53] rely on extracting appearance information from the object pixels using hand-crafted features like HOG and SIFT [7, 48]. We choose to learn the feature extraction directly by using a convolutional pipeline that can be trained fully end-to-end on a large amount of data.

Network Inputs: At each frame, we feed the network a pair of crops from the image sequence. The first crop is centered at the object's location in the previous image, whereas the second crop is in the *same* location, but in the *current* image. The crops are each padded to be twice the size of the object's bounding box to provide the network with context. If the bounding box at frame j had centers (X_c^j, Y_c^j) and width and height W^j, H^j, both crops would be centered at (X_c^j, Y_c^j) with width and height $2W^j$ and $2H^j$. This is illustrated by the colored and dotted boxes in Figure 2. By feeding a pair of crops, the network can directly compare differences in the two frames and learn how motion affects the image pixels. The crops are warped to be 227x227 pixels before being input into the network. We experimentally determined that preserving the aspect ratio of the source images hurts performance because it forces the network to directly regress the aspect ratio rather than regress changes to the ratio.

Skip Connections: The hierarchical structure of convolutional networks extracts different levels of information from different layers [69]; the lowest layers of image classification networks output features like edge maps, whereas the deeper layers capture high-level concepts such as animal noses, eyes, and ears [69]. Rather than only using the outputs from the last layer of the network, we represent the object's appearance using low, mid, and high level features. We use skip connections when spatial resolution decreases to give the network a richer appearance model. In this way, the network can model that it's tracking a person (high level concept) wearing a red (low level concept) shirt rather than a person wearing a blue shirt.

The skip connections are each fed through their own $1 \times 1 \times C$ convolutional layers where C is chosen to be less than the number of input channels. This reduces the dimensionality of the layers with higher spatial resolutions to keep computational cost low. As the spatial resolution is halved, C is doubled. All skip connection outputs and the final output are concatenated together and fed through a final fully-connected layer to reduce the dimensionality of the embedding space that feeds into the recurrent pipeline.

3.2. Recurrent Specifications

Recurrent networks tend to be difficult to train [55]. They can take many more examples to converge than feed-forward networks. We present a method of training a recurrent tracking network which translates the image embedding into an output bounding box while simultaneously updating the internal appearance and motion model. We also describe techniques that lead to faster convergence and better-performing networks.

Recurrent Structure: Using the prior work of Greff *et al.* [23], we opt for a two-layer, factored LSTM (the visual features are fed to both layers) with peephole connections. Both LSTM layers have 1024 units each. We find this outperforms a single layer LSTM even given a deeper convolutional network and larger embedding space. The two layer LSTM is likely able to capture more complex object transformations, and remember longer term relationships than the single layer LSTM. The exact formulation is shown in Equations 1-6 where t represents the frame index, x^t and y^{t-1} are the input and previous output (or recurrent) vectors respectively, $\mathbf{W}$, $\mathbf{R}$, and $\mathbf{P}$ are weight matrices for the input, recurrent, and peephole connections respectively, b is the bias vector, h is the hyperbolic tangent function, σ is the sigmoid function, and $\odot$ is point-wise multiplication. A forward pass produces both an output vector y^t, which is used to regress the current coordinates, and the cell state c_t, which holds important memory information. Both y_t and c_t are fed into the following forward pass, allowing for information to propagate forward in time.

$$z^t = h(\mathbf{W}_z x^t + \mathbf{R}_z y^{t-1} + b_z) \quad \text{LSTM input (1)}$$

$$i^t = \sigma(\mathbf{W}_i x^t + \mathbf{R}_i y^{t-1} + \mathbf{P}_i c^{t-1} + b_i) \quad \text{input gate (2)}$$

$$f^t = \sigma(\mathbf{W}_f x^t + \mathbf{R}_f y^{t-1} + \mathbf{P}_f c^{t-1} + b_f) \quad \text{forget gate (3)}$$

$$c^t = i^t \odot z^t + f^t \odot c^{t-1} \quad \text{cell state (4)}$$

$$o^t = \sigma(\mathbf{W}_o x^t + \mathbf{R}_o y^{t-1} + \mathbf{P}_o c^t + b_o) \quad \text{output gate (5)}$$

$$y^t = o^t \odot h(c^t) \quad \text{LSTM output (6)}$$

The output and cell state vectors update as the object appearance changes. Figure 3 shows a t-SNE [49] plot of the LSTM states our tracker produces for each frame from the the VOT 2014 [38] videos. Because the LSTM states are initialized to 0 at the start of each video, the embeddings of the first few frames from each track are clustered together. As each video progresses, the LSTM state is transformed, resulting in many long, thin paths that follow the ordering of the frames in the original video. Certain points in the sequences with significant occlusion are circled, demonstrating that are embedding does not change drastically during occlusions even though the image pixels look quite different.

Network Outputs: The second LSTM's outputs are fed into a final fully-connected layer with four output values,

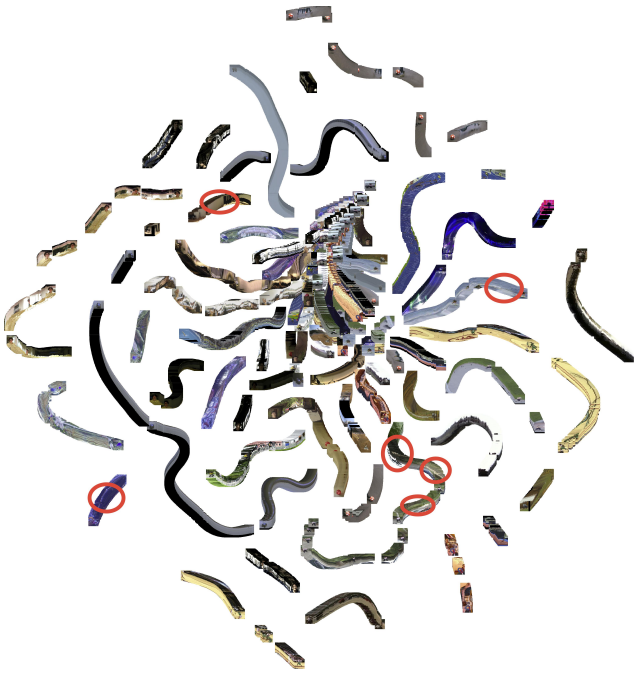

Figure 3. t-SNE embedding of LSTM states from VOT 2014 [38] data. The cell and output states are concatenated together to form the feature vector. The states form paths in the embedding indicating that as the images change during a video, the LSTM states change in a similar fashion. Circled portions of the embedding indicate occlusion.

representing the top left and bottom right corners of the object box in the crop coordinate frame. By regressing these coordinates, we can directly handle size and aspect ratio changes. Similar to [27], we use an L1 loss on the output values to encourage exact matches the ground truth and limit potential drift.

Unrolling during training: Recurrent networks generally take many more iterations to converge than comparable feed-forward networks. This is likely because the inputs are all fed in sequentially, and then one or many outputs and losses are produced, as in [14]. This means the loss must propagate through many noisy intermediate states, causing the gradients to fluctuate and often not be useful for convergence. However, for tracking, each input is directly paired with an immediate output. Thus, we can use a training curriculum that begins with few unrolls, and slowly increases the time horizon that the network sees to teach it longer-term relationships. Without the shorter unroll step, the network may take exponentially longer to train, or may simply never converge. Using this curriculum, we do not find it necessary to clip gradients. Specifically, we initially train the network with only two unrolls and a mini-batch size of 64. After the loss plateaus, we double the number of unrolls and halve the mini-batch size until a maximum unroll of 32 timesteps and a mini-batch size of 4.

Learning to Fix Mistakes: Recurrent networks are often trained by feeding ground truth outputs into the future timesteps rather than the network's own predictions [18, 61]. However, if we always provide the network with ground-truth crops, at test time it quickly accumulates more drift than it has ever encountered, and loses track of the object. To counteract this, we employ a regime that initially relies on ground-truth crops, but over time the network uses its own predictions to generate the next crops. We initially only use the ground truth crops, and as we double the number of unrolls, we increase the probability of using predicted crops to first 0.25, then subsequently 0.5 and 0.75.

3.3. Training Procedure

We use a combination of real and synthetic data to train our deep network. This results in our tracker being able to work on a large variety of object types, allowing us to successfully track across multiple datasets.

Training from Video Sequences: We train Re^3 on two large object tracking datasets: the training set from the ILSVRC 2016 Object Detection from Video dataset (Imagenet Video) [57] and the Amsterdam Library of Ordinary Videos 300++ (ALOV) [58]. In its training set alone, Imagenet Video provides 3862 training videos with 1,122,397 images, 1,731,913 object bounding boxes, and 7911 unique object tracks. This is by far the largest object tracking dataset we are aware of, however it only contains videos for 30 object categories. ALOV consists of 314 videos. We do not use the 7 videos that also occur in VOT 2014 [38] in order to avoid training on the test set. The remaining dataset comprises 307 videos and 148,319 images, each with a single object.

Training from Synthetic Sequences: Due to the large variety of objects labeled in 'object detection in image' datasets, we construct synthetic videos from still images to show the network new types of objects. We use images from the Imagenet Object Detection dataset to fill this role [57]. We discard objects that are less than 0.1^2 of the total image area due to lack of detail, resulting in 478,807 object patches.

To generate simulated data, we randomly sample over all images for an object to track. We use random patches from the same image as occluder patches. The full image serves as the background for the scene. The object, background, and occluders are taken from the same image in order to keep our simulated images close to the real image manifold. We then simulate tracks for the object and occluders, at each timestep modifying an initial speed, direction, and aspect ratio with Gaussian noise. Sample tracks can be found in the supplemental material. This data adds diversity to the types of objects that the network sees, as categories like "person," which are common in many tracking datasets, are absent in Imagenet Video [57].

Tracking at test time: To generate test-time predictions,

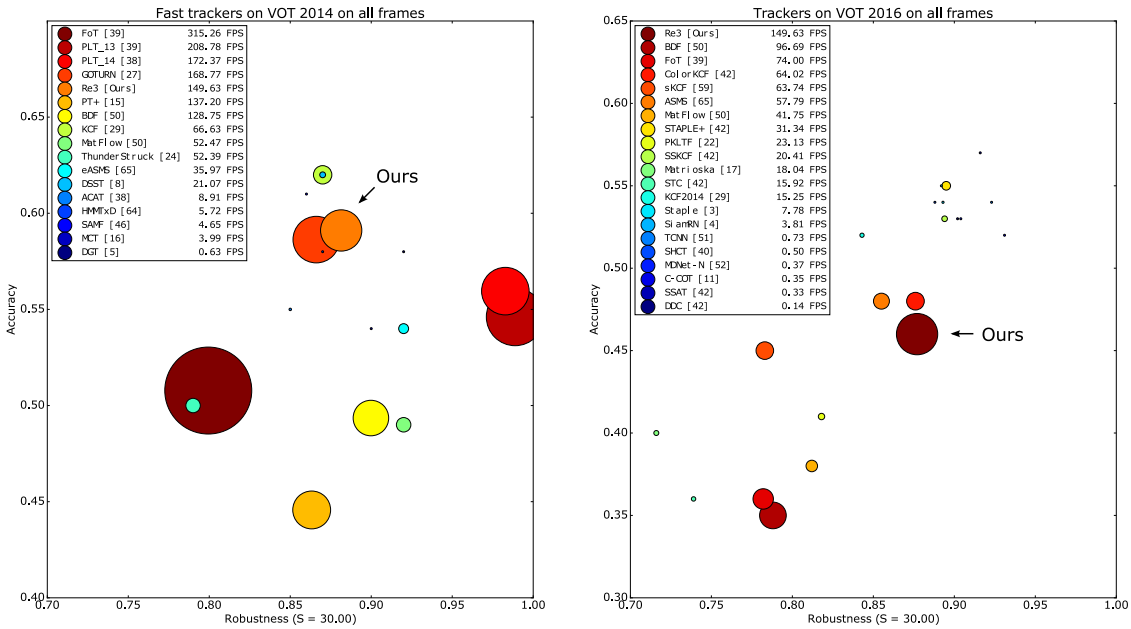

Figure 4. We compare Re^3 to other trackers on the VOT 2014 [38] and VOT 2016 [42] test suites. The size of the point indicates the speed of the tracker. Those below 3 FPS are enlarged to be visible. Speeds are taken directly from the VOT 2014 [38] and VOT 2016 [42] result reports. The VOT authors have stated that speed differences between years can be due to different code, different machines, and other confounding factors. Trackers in Plot A: [39, 39, 38, 27, 15, 50, 29, 50, 24, 65, 8, 38, 64, 46, 16, 5]. Trackers in Plot B: [50, 39, 42, 59, 65, 50, 42, 22, 42, 17, 42, 29, 3, 4, 51, 40, 52, 11, 42, 42]

we feed crops to the network from each sequential frame. After every 32 iterations, we reset the LSTM state. This is necessary because we train on sequences with a maximum length of 32 frames, and without this reset, the LSTM parameters tend to diverge from values the network has seen before. Rather than resetting the LSTM state to all zeros, we use the output from the first forward pass. This maintains an encoding of the tracked object, while allowing us to test on sequences much longer than the number of training unrolls.

Implementation Details: We use Caffe [32] to train and test our networks. Unless otherwise noted, we use the CaffeNet convolutional pipeline initialized with the pretrained weights for our convolutional layers. The skip connections occur after "norm1," "norm2," and "conv5," with 16, 32, and 64 channels respectively. Each skip layer has a PReLU nonlinearity [25]. We initialize all new layers with the MSRA initialization method [25]. We use the the ADAM gradient optimizer [37] with the default Caffe momentum and weight decay and an initial learning rate of 10^{-5}, which we decrease to 10^{-6} after 10,000 iterations and continue for approximately 200,000 iterations. All layers, including the pretrained ones, are updated with this learning rate. During training, we randomly mirror entire tracks with probability 0.5. All tests were carried out using an Intel Xeon CPU E5-2696 v4 @ 2.20GHz and an Nvidia Titan X (Pascal). For timing purposes, we ignore disk read speeds as they are independent of the tracking algorithm used. We plan on releasing a version of our tracker in both Caffe and Tensorflow along with pretrained weight files.

4. Experiments

We compare Re^3 to other tracking methods on several popular tracking datasets in terms of both overall performance and robustness to occlusion. On all datasets, we are among the fastest, most accurate, and most robust trackers. We initially demonstrate our effectiveness by testing on a standard tracking benchmark, the Visual Object Tracking 2014 and 2016 (VOT 2014 and VOT 2016) challenges [38, 42], where we outperform other real-time trackers and are competitive with other deep methods. Next, we show results on the ILSVRC 2016 Object Detection from Video challenge (Imagenet Video) [57] comparing with other real-time trackers. We then examine our performance during occlusion with specific experiments on occluded data. Finally, we perform an ablation study to understand the contributions of each part to the overall success of the method.

4.1. VOT 2014 and 2016

The VOT 2014 and 2016 object tracking test suite [38, 42] consists of 25 and 60 videos respectively made with the explicit purpose of testing trackers. Many of the videos contain difficulties such as large appearance change, heavy occlusions, and camera motion. Trackers are compared in terms of accuracy (how well the predicted box matches with

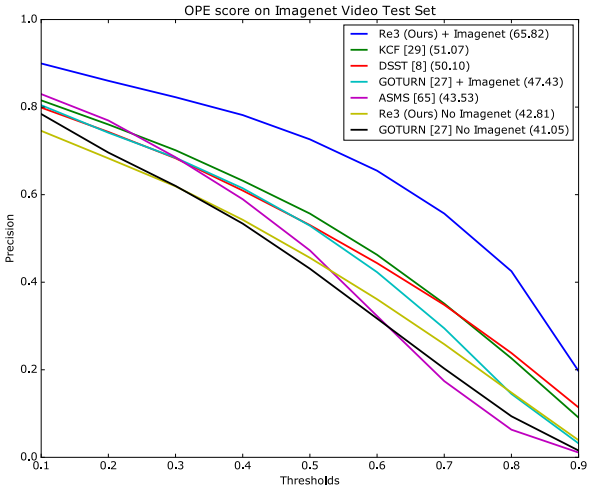

Figure 5. Various real-time trackers evaluated on the Imagenet Video test set [57]. Area under the curve (AUC) is shown for each method. Trackers in Plot: [29, 8, 27, 65] Code provided by [63, 21, 27, 65] respectively.

the ground truth) and robustness (how infrequently a tracker is reset). More details about these criteria can be found in [38]. Figure 4 compares Re^3 with other trackers submitted to the VOT 2014 and 2016 challenges as well as with Held *et al*. [27, 38, 42]. We show the 10 fastest trackers as well as the 10 most accurate trackers from each year.

Figure 4A shows our full model trained using all of the available training data. Our high robustness indicates our model's ability to retain tracks for longer periods than many others, most likely due to the LSTM's ability to directly model temporal changes. More robust methods such as PLT_13 and PLT_14 [38, 39] are less accurate, indicating that their bounding boxes may be less useful.

Figure 4B compares our results against more modern trackers on the more difficult VOT 2016 test set [42]. For training this model, we omit the ALOV data since there is a large overlap between the two video sets. We later explore the effect this has on our network's performance in the ablation analysis (model G in Table 1). Re^3 is 450x faster than the best methods [11, 42], while scoring only 20% and 5% lower in terms of relative accuracy and robustness. On both datasets, Re^3 offers an attractive trade-off of speed, accuracy, and robustness.

4.2. Imagenet Video

The Imagenet Video validation set consists of 1309 individual tracks and 273,505 images [57]. It is the largest dataset we test on, and it offers significant insights into the success cases and failure cases of our tracker. We use Imagenet Video to evaluate our performance against other open-source real-time trackers. Each tracker is initialized with the first frame of each test sequence and is not reset upon losing

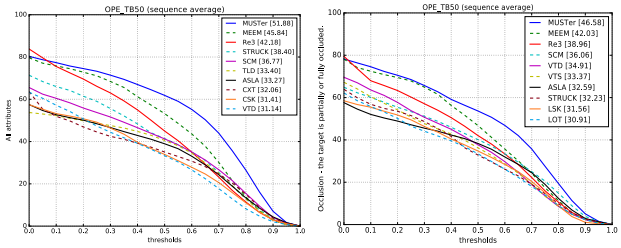

Figure 6. Evaluation on the OTB benchmark [68]. We examine performance both overall (left) and during occluded frames (right) using the One Pass Evaluation (OPE) criterion explained in [68]. The legend shows area under the curve (AUC) for each method. Relative to other trackers, we suffer a smaller loss in accuracy due to occlusion. Trackers in Plot A: [30, 70, 24, 72, 35, 31, 13, 28, 43]. Trackers in Plot B: [30, 70, 72, 43, 44, 31, 24, 47, 54]

track. Each individual bounding box is evaluated against the ground truth for that track at various IOU thresholds. Figure 5 shows our method outperforming other real-time trackers over all thresholds by a wide margin, though only our method and GOTURN [27] + Imagenet were trained with the Imagenet Video training set. We also train a version of our network without using the Imagenet Video training data, only using a combination of ALOV [58] and simulated data. This performs significantly worse, most likely because LSTMs tend to take more data to train than comparable feed forward methods. With sufficient training data, our method outperforms other methods trained on the same data.

4.3. Online Object Tracking benchmark

The Online Object Tracking benchmark (OTB) [68] is a widely used benchmark in tracking literature consisting of 50 challenging tracking videos of various objects. The One Pass Evaluation (OPE) criteria on OTB is equivalent to the evaluation we perform on the Imagenet Video. In Figure 6, we show results competitive with other modern baselines, yet we operate at a much faster frame rate. This experiment also illustrates that our method performs significantly better when given training data of the same category of object, as in the Imagenet experiment, versus arbitrary objects. Our method implicitly learns the dynamics of object categories using the LSTM as well as high level image cues from the late CNN features that offer performance boosts when testing on the same objects. When we compare this to novel objects that the network was not trained on we see significantly higher drift.

4.4. Robustness to Occlusion

We present two additional experiments showing that Re^3 performs comparatively well during occlusions. LSTMs can implicitly learn to handle occlusions because the structure of an LSTM can ignore information via the input and

			VOT 2014				Imagenet Video			
	Network Structure and Training Method	**Speed (FPS)**	**Accuracy**	**# Drops**	**Robustness**	**Average**	**Accuracy**	**# Drops**	**Robustness**	**Average**
A	Feed Forward Network (GOTURN) [27]	168.77	0.61	35	0.90	0.756	0.55	471	0.95	0.750
B	A + Imagenet Video Training	168.77	0.55	41	0.89	0.718	0.56	367	0.96	0.760
C	One Layer LSTM	**213.27**	0.48	67	0.82	0.651	0.49	738	0.92	0.706
D	C + Self-training	**213.27**	0.57	43	0.88	0.726	0.6	450	0.95	0.776
E	D + Simulated Data	**213.27**	0.6	38	0.89	0.747	0.65	359	0.96	0.806
F	E + Skip Layers	160.72	0.62	29	**0.92**	0.769	0.69	320	**0.97**	0.828
G	Full Model (F with two LSTM layers)	149.63	0.66	29	**0.92**	0.789	0.68	257	**0.97**	0.826
H	Full Model No ALOV	149.63	0.6	28	**0.92**	0.761	**0.71**	**233**	**0.97**	**0.842**
I	Full Model No Imagenet Video	149.63	0.58	61	0.82	0.700	0.52	1096	0.88	0.700
J	Full Model with GoogleNet [60] conv layers	77.29	**0.68**	**27**	**0.92**	**0.802**	0.69	274	**0.97**	0.830

Table 1. Ablation Study. Average represents the arithmetic mean of accuracy and robustness, providing a single score to each method. Results on VOT 2014 [38] differ slightly from the VOT test suite, as they consider bounding boxes with a rotation angle, and we take the outermost points on these boxes as the ground truth labels.

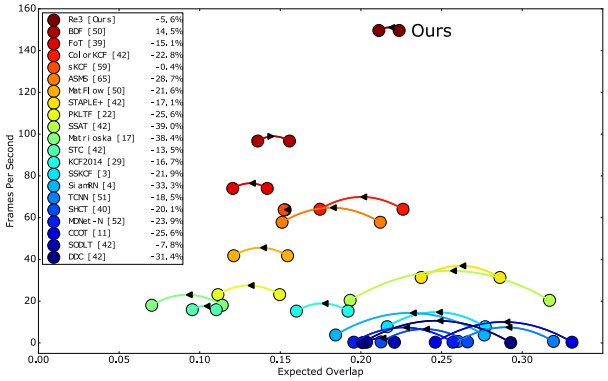

Figure 7. Expected overlap of various trackers compared between all frames and occluded frames. The arrow indicates that BDF [50] improves slightly during occlusion whereas all other methods degrade. Trackers in Plot: [50, 39, 42, 59, 65, 50, 42, 22, 42, 17, 42, 29, 3, 4, 51, 40, 52, 11, 42, 42]

forget gates. This contrasts many other methods which assume all observations are useful, and may update their internal representation to include the occluder's appearance. In these experiments, we compare both the quality of track during occlusions as well as the difference in performance between overall scores and scores during occluded frames.

First, we examine our performance on the VOT 2016 test set [42]. Figure 7 shows the expected overlap measure of the same trackers from Figure 4B. Expected overlap represents the trackers' accuracy and robustness as a single number by performing many trials on subsets of the original data (more details available in [41]). Each tracker has two points on the graph: the first for overall expected overlap, and the second for expected overlap during occlusion. Re^3's performance degrades slightly during occlusions, but many of the other trackers drop in accuracy by more than 25%. sKCF [59] also barely changes, and BDF [50] actually improves during occlusions.

We also evaluate how our tracker performance degrades during occlusions across various IOU thresholds on the Online Object Tracking benchmark (OTB) [68]. Similar to the previous experiment, Figure 6 compares the performance of trackers during all frames, and only during occluded frames. Again, we suffer a smaller loss in accuracy of 7.6 in relative percentage compared to other top methods (MUSTer [30] 10.2%, MEEM [70] 8.3%, STRUCK [24] 16.1%). The performance on both datasets under occlusion illustrate that our LSTM-based method offers significant robustness to occlusion - one of the most difficult challenges in object tracking.

4.5. Ablation Study

Table 1 examines how various changes to the network affect the speed and performance of Re^3 on the VOT 2014 and Imagenet Video test sets [38, 57]. The difference between model A and C is that model A has three fully-connected layers with 4096 outputs each, whereas C has one fully-connected layer with 2048 outputs, and one LSTM layer with 1024 outputs. Despite the small change, simply adding an LSTM to an existing tracker without any modification in the training procedure hinders performance. Learning to correct previous mistakes and prevent drift (model D), or self-training, is clearly necessary when training a recurrent tracker. Other modifications tend to add slight improvements in both accuracy and robustness. At the expense of speed, we can attain even higher numbers. Model J uses the GoogleNet [60] architecture to embed the images, but is twice as slow. Model H, which was trained only on Imagenet Video [57] and simulated data, shows that by training on a fixed set of classes, performance improves on those classes but drops significantly on new objects (VOT 2014 [38]). Model I illustrates the need for a large training dataset, which seems especially important in terms of robustness.

5. Conclusion

In this paper, we presented the first algorithm that uses a recurrent neural network to track generic objects in a variety of natural scenes and situations. Recurrent models offer a new, compelling method of tracking due to their ability to learn from many examples offline and to quickly update online when tracking a specific object. Because they are end-to-end-trainable, recurrent networks can directly learn robustness to complex visual phenomena such as occlusion and appearance change. Our method demonstrates in-

creased accuracy, robustness, and speed over comparable trackers, especially during occlusions. We showed how to efficiently and effectively train a recurrent network to learn from labeled videos and synthetic data. Ultimately we have shown that recurrent neural networks have great potential in the fast object tracking domain.

References

[1] Large scale visual recognition challenge 2016 (ilsvrc2016), 2016.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009. 2

[3] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409, 2016. 6, 8

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pages 850–865. Springer, 2016. 2, 3, 6, 8

[5] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, and S. Z. Li. Robust deformable and occluded object tracking with dynamic graph. *IEEE Transactions on Image Processing*, 23(12):5497–5509, 2014. 6

[6] J.-W. Choi, D. Moon, and J.-H. Yoo. Robust multi-person tracking for real-time intelligent video surveillance. *ETRI Journal*, 37(3):551–561, 2015.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. 4

[8] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014. 6, 7

[9] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 2

[10] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 58–66, 2015. 3

[11] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488. Springer, 2016. 6, 7, 8

[12] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014. 2

[13] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1177–1184. IEEE, 2011. 7

[14] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015. 5

[15] S. Duffner and C. Garcia. Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects. In *Proceedings of the IEEE international conference on computer vision*, pages 2480–2487, 2013. 2, 6

[16] S. Duffner and C. Garcia. Using discriminative motion context for online visual object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(12):2215–2225, 2016. 2, 6

[17] M. Edoardo Maresca and A. Petrosino. The matrioska tracking algorithm on ltdt2014 dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 706–711, 2014. 2, 4, 6, 8

[18] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015. 5

[19] Q. Gan, Q. Guo, Z. Zhang, and K. Cho. First step toward model-free, anonymous object tracking with recurrent neural networks. *arXiv preprint arXiv:1511.06425*, 2015. 3

[20] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[21] Georg Nebehay. Dsst: Discriminative scale space tracker. 7

[22] A. González, R. Martín-Nieto, J. Bescós, and J. M. Martínez. Single object long-term tracker for smart

control of a ptz camera. In *Proceedings of the International Conference on Distributed Smart Cameras*, page 39. ACM, 2014. 6, 8

[23] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015. 4

[24] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2016. 2, 6, 7, 8

[25] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015. 6

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016. 1

[27] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference Computer Vision (ECCV)*, 2016. 2, 3, 5, 6, 7, 8

[28] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012. 7

[29] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 2, 6, 7, 8

[30] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 749–758, 2015. 7, 8

[31] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1822–1829. IEEE, 2012. 7

[32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6

[33] S. E. Kahou, V. Michalski, and R. Memisevic. Ratm: Recurrent attentive tracking model. *arXiv preprint arXiv:1510.08660*, 2015. 3

[34] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012. 2

[35] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012. 7

[36] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *arXiv preprint arXiv:1604.02532*, 2016. 3

[37] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[38] M. Kristan, R. Pflugfelder, A. Leonardis, et al. The visual object tracking vot2014 challenge results. *The Visual Object Tracking Workshop 2014 at ECCV2014, 2014*, 2014. 2, 4, 5, 6, 7, 8

[39] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al. The visual object tracking vot2013 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 98–111, 2013. 2, 6, 7, 8

[40] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al. The visual object tracking vot2013 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 98–111, 2013. 6, 8

[41] M. e. a. Kristan. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–23, 2015. 1, 2, 8

[42] M. e. a. Kristan. The visual object tracking vot2016 challenge results. In *Proceedings of the IEEE European Conference on Computer Vision workshops*, 2016. 2, 6, 7, 8

[43] J. Kwon and K. M. Lee. Visual tracking decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1269–1276. IEEE, 2010. 7

[44] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1195–1202. IEEE, 2011. 7

[45] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Long-term tracking through failure cases. In *Proceed-*

ings of the IEEE International Conference on Computer Vision Workshops, pages 153–160, 2013. 2

[46] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *European Conference on Computer Vision*, pages 254–265. Springer, 2014. 2, 6

[47] B. Liu, J. Huang, L. Yang, and C. Kulikowsk. Robust tracking using local sparse appearance model and k-selection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1313–1320. IEEE, 2011. 7

[48] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 4

[49] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 4

[50] M. E. Maresca and A. Petrosino. Clustering local motion estimates for robust and efficient object tracking. In *European Conference on Computer Vision*, pages 244–253. Springer, 2014. 6, 8

[51] H. Nam, M. Baek, and B. Han. Modeling and propagating cnns in a tree structure for visual tracking. *arXiv preprint arXiv:1608.07242*, 2016. 6, 8

[52] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 3, 6, 8

[53] G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 862–869. IEEE, 2014. 2, 4

[54] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan. Locally orderless tracking. *International Journal of Computer Vision*, 111(2):213–228, 2015. 7

[55] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013. 4

[56] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[57] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5, 6, 7, 8

[58] A. W. Smeulders, D. M. 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):1442–1468, 2014. 5, 7

[59] A. Solis Montero, J. Lang, and R. Laganiere. Scalable kernel correlation filter with sparse feature integration. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 24–31, 2015. 6, 8

[60] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 8

[61] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015. 5

[62] T. Vojir. Robust scale-adaptive mean-shift for tracking.

[63] T. Vojir. Tracking with kernelized correlation filters. 7

[64] T. Vojir, J. Matas, and J. Noskova. Online adaptive hidden markov model for multi-tracker fusion. *Computer Vision and Image Understanding*, 153:109–119, 2016. 6

[65] T. Vojir, J. Noskova, and J. Matas. Robust scale-adaptive mean-shift for tracking. In *Scandinavian Conference on Image Analysis*, pages 652–663. Springer, 2013. 6, 7, 8

[66] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015. 3

[67] A. Wendel, S. Sternig, and M. Godec. Robustifying the flock of trackers. In *16th Computer Vision Winter Workshop. Citeseer*, page 91. Citeseer, 2011.

[68] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013. 7, 8

[69] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014. 4

[70] J. Zhang, S. Ma, and S. Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *European Conference on Computer Vision*, pages 188–203. Springer, 2014. 7, 8

[71] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015. 1

[72] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1838–1845. IEEE, 2012. 7