Optical Flow in Deep Visual Tracking

Mikko Vihlman, Arto Visala
Department of Electrical Engineering and Automation, Aalto University
PO Box 15500, FIN-00076 Aalto, Finland
{mikko.vihlman, arto.visala}@aalto.fi

Abstract

Single-target tracking of generic objects is a difficult task since a trained tracker is given information present only in the first frame of a video. In recent years, increasingly many trackers have been based on deep neural networks that learn generic features relevant for tracking. This paper argues that deep architectures are often fit to learn implicit representations of optical flow. Optical flow is intuitively useful for tracking, but most deep trackers must learn it implicitly. This paper is among the first to study the role of optical flow in deep visual tracking. The architecture of a typical tracker is modified to reveal the presence of implicit representations of optical flow and to assess the effect of using the flow information more explicitly. The results show that the considered network learns implicitly an effective representation of optical flow. The implicit representation can be replaced by an explicit flow input without a notable effect on performance. Using the implicit and explicit representations at the same time does not improve tracking accuracy. The explicit flow input could allow constructing lighter networks for tracking.

Introduction

The goal of single-target visual tracking is to find the location of a specific object in subsequent video frames given the location of the object in the beginning of the video. Visual tracking has several applications in computer vision and automation. For instance, surveillance systems use tracking to automatically follow certain individuals whereas autonomous vehicles need tracking to follow the road, traffic signs and obstacles. Research often focuses on tracking either objects of a given class, e.g. people or vehicles, or generic objects of any class. This paper concentrates on single-target tracking of generic objects.

Despite extensive research, visual tracking remains an active and challenging field of study. Real-world videos involve many complicating factors such as occlusions, nonrigid transformations, movements of the camera and changes in illumination. Learning to track is difficult since the goal is to track previously unseen objects in new environments; a pretrained tracker is initialized using only the information present in the beginning of the sequence.

During the past years, visual tracking has been dominated by correlation filters and deep trackers. Discriminative correlation filters model the target object based on a single frame and adapt the learned representation in online fashion based on subsequent frames during the tracking process. They are traditionally based on handcrafted or shallow features but can also be built on deep features. Correlation filters are efficient but often not discriminative enough in complex conditions. Deep models trained end-to-end for tracking can be more discriminative. Deep neural networks have been used successfully in applications such as image recognition (Krizhevsky, Sutskever, and Hinton 2012), object detection (Girshick et al. 2014; Ren et al. 2015), semantic segmentation (Long, Shelhamer, and Darrell 2015) and action recognition (Simonyan and Zisserman 2014). Recently, large datasets like the ILSVRC (Russakovsky et al. 2015) dataset for object detection in video have enabled large-scale offline training to learn generic features for tracking.

One approach to ease the learning process is to present the data in a new way using traditional methods of computer vision. For instance, (Simonyan and Zisserman 2014) has shown that feeding the network with the optical flow between consecutive frames instead of raw frames can notably improve the performance of deep networks in action recognition. The improvement arguably results from the fact that the network does not need to implicitly learn how to compute optical flow or to understand that the flow information is useful. In this paper, we argue that optical flow could similarly be useful for deep visual tracking.

The tracking literature focuses heavily on the final performance of the trackers, often providing little insight into why the trackers perform well. Instead, this paper deliberately concentrates on analyzing the learned representations. This is highly important as it can increase the understanding of how existing trackers work and reveal sources of good tracking performance. In this paper, the focus is on the role of optical flow. The analysis is done by modifying the architecture of the GOTURN tracker (Held, Thrun, and Savarese 2016), one typical approach to deep visual tracking. The goal is to find out if the tracker uses implicit representations of optical flow, how useful the representations are, and how using optical flow more explicitly affects the tracking process. In addition to being one of the few papers analyzing the representations learned by deep trackers, this paper is among the first to explicitly use optical flow as input for deep visual tracking. The results highlight the usefulness of optical flow and the capability of neural networks to implicitly learn effective representations of the flow information.

Related Work

Prior to deep trackers, visual tracking was dominated by approaches based on correlation filters. MOSSE (Bolme et al. 2010) builds trackers directly from pixel values, whereas the kernelized correlation filter (Henriques et al. 2015) uses HOG features. More developed methods include Staple (Bertinetto et al. 2016a), SAMF (Danelljan et al. 2014), SRDCF (Danelljan et al. 2015) and BACF (Galoogahi, Fagg, and Lucey 2017). Correlation filters have also been learned from features of convolutional neural networks (e.g. (Ma et al. 2015)). This makes the tracker more discriminative, at the cost of computational complexity. Even correlation filters built on deep features are in a way handcrafted since there is typically no finetuning to the features or the filters. Recently, (Valmadre et al. 2017) has formulated correlation filter as a differentiable layer that allows learning deep features coupled to the correlation filter.

Traditional tracking datasets such as the Object Tracking Benchmark (Wu, Lim, and Yang 2015), the Visual Object Tracking (VOT) challenge (see e.g. (Kristan et al. 2015)) and the ALOV++ dataset (Smeulders et al. 2014) contain only tens or hundreds of videos. In effect, many deep trackers use online training, i.e. learning during the tracking process. For instance, MDNet (Nam and Han 2016) takes samples around the previous location and estimates how likely they represent the target object. The network is finetuned online using positive and negative samples. FCNT (Wang et al. 2015) learns two branches online to produce foreground heat maps from convolutional features. DSiam (Guo et al. 2017) applies target appearance variation and background suppression transformations on top of a Siamese network. CREST (Song et al. 2017) reformulates the correlation filter as a layer on top of a Siamese network and uses residual learning to reduce model degradation during online updates. VITAL (Song et al. 2018) uses adversarial learning to augment positive samples and to reduce the effect of easy negative samples.

Online training allows the network to track objects that it has never seen before and to adapt to changing conditions. However, the quality of online data for training is typically low; there is the risk of learning useless or misleading features. Moreover, the additional learning process required by online learning typically reduces the tracking speed. Instead of or in addition to online learning, many deep trackers are based on offline training. This allows learning generic features and could remove the need for costly online updates.

There are many ways to implement offline learning for tracking. GOTURN (Held, Thrun, and Savarese 2016) processes the current and previous frames with convolutional layers and applies a branch of fully-connected layers to the concatenated convolutional features to estimate the bounding box of the target object. SiamFC (Bertinetto et al. 2016b) computes the cross-correlation between the convolutional features of the current frame and object template. The network runs a single forward pass to estimate how similar each location of the current frame is to the template. Siamese-RPN (Li et al. 2018) uses a region-proposal network containing template and detection branches that can be trained offline for tracking. UCT (Zhu et al. 2017) learns feature extractors and a correlation filter using offline training and updates the model using online training.

Optical flow measures the displacement of pixels from one frame to another. It is intuitively useful for visual tracking; however, deep trackers seldom use the flow information explicitly. The SINT tracker (Tao, Gavves, and Smeulders 2016) uses optical flow as a guide during tracking to limit the size of the search region. Thus, the flow information serves mainly to improve efficiency or robustness without modifying the representations learned by the network.

Although typical deep trackers do not explicitly use optical flow, they can learn implicit representations of the flow. Many trackers loosely resemble the FlowNet architectures (Dosovitskiy et al. 2015) that are used for estimating the optical flow with convolutional neural networks. For instance, the fact that GOTURN (Held, Thrun, and Savarese 2016) concatenates convolutional outputs prior to estimating the location of the object makes the operation fairly similar to FlowNetSimple (Dosovitskiy et al. 2015), albeit for different structure after concatenation. The way SiamFC (Bertinetto et al. 2016b) uses cross-correlation to measure the similarity between the current frame and object template loosely reminds the operation of FlowNetCorr (Dosovitskiy et al. 2015). These examples suggest that at least some representation of optical flow is useful for deep visual tracking.

To our knowledge, this paper is among the first to explicitly use optical flow as input for deep visual tracking. Recently, (Zhu et al. 2018) uses optical flow to warp and aggregate historical feature maps with current ones to learn correlation filters for tracking. The framework allows joint learning of deep optical flow and tracking. Our paper differs from (Zhu et al. 2018) in several ways. First, we use traditional optical flow instead of FlowNet (Dosovitskiy et al. 2015). Second, to make the tracking process more efficient and to simplify our analysis, our network does not involve any online learning. Third, our method uses a traditional deep architecture instead of correlation filters. Finally, our architecture does not specify in any way how the network should utilize the flow information.

Methods

Basic Principle

This paper builds directly on the GOTURN tracker (Held, Thrun, and Savarese 2016). GOTURN processes the current and previous frames using identical branches of convolutional layers. The convolutional outputs are fed into a branch of fully-connected layers that outputs the bounding box of the object in the current frame. As shown in Figure 1, this paper uses an additional convolutional branch that processes the optical flow between the two frames. Modifying the architecture by ignoring some of the input branches (more details shortly) allows identifying implicit representations of optical flow and analyzing the effect of using flow information more explicitly.

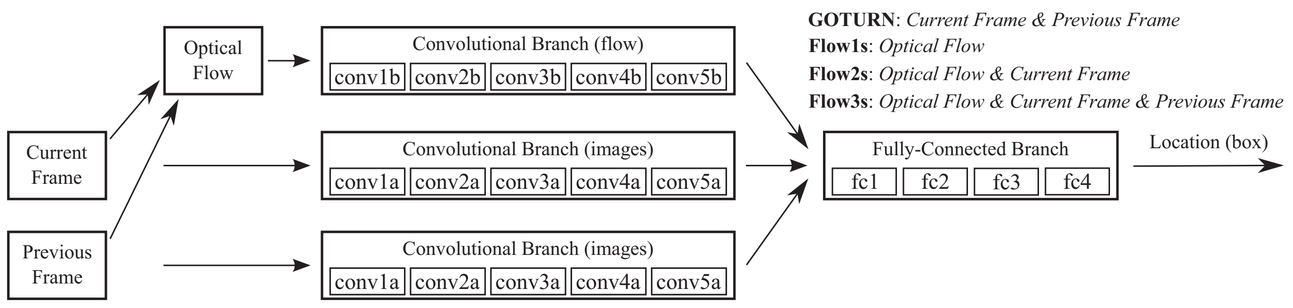
GOTURN: Current Frame & Previous Frame
Flow1s: Optical Flow
Flow2s: Optical Flow & Current Frame
Flow3s: Optical Flow & Current Frame & Previous Frame
Optical Flow
Current Frame
Previous Frame
Convolutional Branch (flow)
conv1b conv2b conv3b conv4b conv5b
Convolutional Branch (images)
conv1a conv2a conv3a conv4a conv5a
Convolutional Branch (images)
conv1a conv2a conv3a conv4a conv5a
Fully-Connected Branch
fc1 fc2 fc3 fc4
Location (box)

Figure 1: The tracking architectures. Depending on the inputs of the network, three different trackers based on optical flow are proposed. All architectures estimate the location (bounding box) of the target object in the current frame based on different sets of inputs. The simplest architecture (Flow1s) bases tracking only on the optical flow between the current and previous frames. The second architecture (Flow2s) accompanies the optical flow input with the current frame. The third architecture (Flow3s) uses all three inputs (optical flow, current frame, previous frame). The frames are cropped and centered around the previous location of the object prior to computing the optical flow. The benchmark model GOTURN (Held, Thrun, and Savarese 2016) bases tracking on the current and previous frames (does not use optical flow explicitly).

The original GOTURN architecture ignores the explicit optical flow input. Given two consecutive frames, GOTURN can use single-frame features and implicitly learned motion features (e.g. optical flow). This paper considers three new architectures: *Flow1s*, *Flow2s* and *Flow3s*, where *Xs* refers to the number of input streams. The Flow1s architecture has a single convolutional branch that processes optical flow; it does not get the current and previous frames. Flow1s allows studying whether explicit optical flow is enough for accurate tracking. The Flow2s architecture has branches for optical flow and the current frame. The Flow1s and Flow2s architectures mostly prevent implicit models of optical flow since the network does not process the previous frame. Thus, they allow replacing the implicit motion representations learned by GOTURN with more explicit flow information. The Flow3s architecture processes the optical flow input and both image frames; thus, it allows building a tracker that uses both implicit and explicit representations of optical flow.

The proposed architectures build on GOTURN for two reasons. First, GOTURN is very simple and intuitive and easily extended in the number of inputs. It provides a good base for incorporating optical flow into a deep tracker. Second, the close similarity allows using GOTURN as a natural benchmark to assess the role of optical flow while changing the input branches. There are also some drawbacks. More recent deep trackers outperform GOTURN; the similarity to GOTURN limits the performance of our trackers. There might be more natural or effective ways to use optical flow. However, the main goal of this paper is to analyze the role of optical flow in deep visual tracking instead of creating best-performing trackers. Thus, a simple architecture is sufficient. Finding the best architecture and improving the tracking performance is left for future research.

Inputs

Minimal processing is applied to the raw images. The frames are cropped and centered around the previous estimated location of the object. Additional margin is added around the bounding box to include some context in the input. Thus,

Layer	Parameters	Notes
Conv 1	3(2)x11x11	96 outputs, stride 4, ReLU
Max Pooling	–	kernel 3x3, stride 2
Local Norm.	–	
Conv 2	96x5x5	256 outputs, stride 1, ReLU
Max Pooling	–	kernel 3x3, stride 2
Local Norm.	–	
Conv 3	256x3x3	384 outputs, stride 1, ReLU
Conv 4	384x3x3	384 outputs, stride 1, ReLU
Conv 5	384x3x3	256 outputs, stride 1, ReLU
Max Pooling	–	kernel 3x3, stride 2

Table 1: The structure of the convolutional branches.

like in GOTURN (Held, Thrun, and Savarese 2016), the object is expected to stay close to its previous location. Margins extending outside the frames are padded with the average color of the dataset. The cropped frames are resized to a fixed size of 227 $\times$ 227 pixels. The average color of the dataset is subtracted from the resized frames to make the average value of the images approximately zero.

The optical flow input is computed between the resized frames (prior to subtracting the mean) using the Farneback method (Farnebäck 2003). It is represented as a two-channel image showing the horizontal and vertical displacement for each pixel. Using a simple and traditional method like the Farneback method is sufficient since the explicit flow input is used mainly to identify and replace implicit representations of optical flow. Aiming for great tracking performance using the best modern methods is left for future research.

Architecture

The networks use the GOTURN architecture (Held, Thrun, and Savarese 2016), except for the adjustments needed for the inputs. Each input present in a given network is processed with a branch of five convolutional layers. The convolutional layers are adapted from CaffeNet (Jia et al. 2014), which is analogous to AlexNet (Krizhevsky, Sutskever, and Hinton 2012). The convolutional branch used for processing the flow input has the same architecture, except that the filters of the first layer process inputs with two (flow) channels instead of three (color) channels. For details of the convolutional branches, see Table 1 and (Jia et al. 2014).

In each network, the final convolutional features are concatenated and fed into a branch of four fully-connected layers. The first three layers each have 4096 outputs. The final layer gives four outputs: the top-left and bottom-right corners of the estimated bounding box. Since the proposed architectures have 1–3 inputs, the number of parameters in the first fully-connected layer varies. Flow2s with two inputs has as many parameters as GOTURN; Flow1s has 50% fewer and Flow3s 50% more.

Some tests are run also using lighter architectures. Optical flow provides a richer representation of the inputs than raw images. It could make the learning process easier and allow using networks of smaller learning capacity. The lighter architectures are equal to the full architectures just presented, except for the structure of the fully-connected branch. In the lighter architectures, the first fully-connected layer is removed. Also, the number of parameters of the original second and third fully-connected layers is decreased to one fourth (1024 outputs instead of 4096). This reduces the number of learned parameters notably and might also make the trackers faster, depending on implementation.

Training

The architectures are fully separate; the parameters are learned offline one network at a time using random pairs of frames. For each input pair, the network being trained estimates the bounding box of the object in the newer frame. The parameters are updated based on the L1 loss between the top-left and bottom-right corners of the estimated and ground-truth bounding boxes.

Training is done using the Stochastic Gradient Descent (SGD) with momentum, mostly like in the original GOTURN paper (Held, Thrun, and Savarese 2016). Pairs of frames are chosen randomly from the annotated sequences. Additional ten synthetic training examples are created from each sampled pair by randomly shifting and scaling the ground-truth bounding box in the newer frame. Training is done for 450,000 iterations using batches of 50 examples.

The original GOTURN tracker (Held, Thrun, and Savarese 2016) was trained using about 300 videos from ALOV++ (Smeulders et al. 2014) and a large set of still images annotated for object detection. The ALOV videos are partly from the same domain as the test videos of the VOT challenge. In this paper, training and validation is done using the ILSVRC (Russakovsky et al. 2015) dataset for object detection in video (ImageNet Video) in order to evaluate the trackers using VOT. There are 3862 videos for training and 555 for validation. The number of tracking sequences is somewhat larger since each video can contain multiple annotated objects. While sampling pairs of frames for training, the distance between the frames is fixed at five frames to modify the boxes using the same settings as in (Held, Thrun, and Savarese 2016). Even though the video dataset is relatively large, we find that it is useful to use also still images during training. The still images are sampled from the same dataset as the real pairs of frames.

The fully-connected layers are initialized with random Gaussian weight parameters and constant bias terms. The convolutional branches of the image inputs are initialized using CaffeNet pretrained for the ILSVRC (Russakovsky et al. 2015) Recognition Challenge. The convolutional branch of the optical flow input requires separate parameters since optical flow is conceptually different from color images. In this paper, the flow branch is initialized either 1) with random Gaussian weights and constant bias terms, or 2) by pretraining for action recognition based on optical flow. The latter option is done by learning the temporal (flow) stream of the two-stream network of (Simonyan and Zisserman 2014). The temporal stream classifies the actions performed in videos based on short sequences of optical flow. To adapt the temporal stream for our purposes, the network of (Simonyan and Zisserman 2014) is replaced by our layers and a single optical flow sample is used as input per example. See (Simonyan and Zisserman 2014) for further details.

Training is repeated multiple times to find the best way to initialize and finetune the convolutional parameters applied to the flow input. After choosing one of the two initialization options, the base learning rate of the parameters applied to the flow input is set to 0, 1e-5 or 1e-6 (fix or finetune). Fully-connected layers are always updated using a base learning rate of 1e-5. The convolutional layers connected to the image inputs have a learning rate of 0; that is, their parameters are fixed. Learning rates of all layers are divided by 10 after every 100,000 iterations.

Implementation

The proposed trackers are implemented using the Caffe deep learning framework (Jia et al. 2014). To make the trackers directly comparable to the GOTURN tracker (Held, Thrun, and Savarese 2016), the implementation builds on the original code of GOTURN. Changes are made to allow using the larger ImageNet Video dataset and to feed the networks with the optical flow input. We also train the original GOTURN within our framework to make the results more comparable.

The inputs of the networks are prepared using CPU, other computations are done on GPU. Optical flow is considered a preprocessing task and run on the CPU. Trackers could be made more efficient by implementing optical flow on GPU. To minimize the time spent on preparing the inputs of the networks, all inputs are sampled and preprocessed once and saved to Lightning Memory-Mapped Databases (LMDBs). The LMDBs are written using JPEG compression to limit the size of the databases. The two-channel flow inputs are saved one channel at a time rescaling each channel linearly to a [0, 255] range (the original ranges are restored during training). In addition, the inputs are prepared only for the first 100,000 iterations and the same LMDBs are used 4.5 times during the training process. Preparing the inputs once before training speeds up training notably and allows using the same random inputs for all networks.

Experiments

The VOT toolkit (Kristan et al. 2015) is used to validate and test the trackers. Following the best practices of VOT, the test videos are used to test only the final selected networks. However, the toolkit can be used to evaluate trackers using a separate set of validation videos. The proposed three architectures are trained with different initialization and finetuning settings using the training videos of ImageNet Video. Validation is done with the VOT toolkit using sequences generated from the validation videos of ImageNet Video to choose the best settings for each architecture.

Final tests are done once for each architecture using the actual VOT test videos. We consider relatively old versions of the dataset (VOT2015 and VOT2016) to make the results comparable to the original GOTURN (Held, Thrun, and Savarese 2016) and to the FlowTrack tracker (Zhu et al. 2018) that also uses optical flow for tracking. This is important since our focus is on understanding the learned representations instead of developing state-of-the-art trackers.

Results

Training and Validation

Table 2 shows the validation results for the architectures. The first two columns show the initialization method and base learning rate for the parameters of the convolutional branch of the flow input. The benchmark GOTURN (Held, Thrun, and Savarese 2016) is trained only once since it does not have a branch for the optical flow. *Overlap* measures the overlap between estimated and real bounding boxes. The VOT toolkit resets the tracker after each failure (zero overlap). *Failures* measures the frequency of failed tracking.

Looking at Table 2, the best results for Flow2s (optical flow and current frame as inputs) and Flow3s (optical flow, current frame and previous frame as inputs) are obtained when the flow branch uses fixed convolutional parameters pretrained for flow-based action recognition. We find that finetuning Flow2s and Flow3s further decreases the value of the loss function during training. However, Table 2 suggests that this results in overfitting as the validation performance of the trackers becomes worse. Thus, for the rest of this paper, the flow branches of the proposed architectures are initialized using parameters pretrained for action recognition and they are not finetuned. Only the fully-connected layers of the networks are learned and finetuned specifically for tracking. Although the performance of Flow1s improves when the flow branch is finetuned, we use the same training settings for all architectures for simplicity and clarity.

Table 2 allows making some remarks about the relative validation performance of the trackers. The Flow1s architecture is the worst tracker regardless of the training settings. Flow2s reaches slightly higher Overlap than GOTURN but also provides somewhat more Failures. Flow3s performs very similarly to GOTURN. Overall, the performance differences between Flow2s, Flow3s and GOTURN are small.

P_0	γ	Method	Overlap	Failures
		GOTURN	0.61	0.26
pretrained	0	Flow1s	0.4	1.21
pretrained	0	Flow2s	0.64	0.38
pretrained	0	Flow3s	0.6	0.26
pretrained	1e-5	Flow1s	0.46	0.66
pretrained	1e-5	Flow2s	0.53	0.43
pretrained	1e-5	Flow3s	0.51	0.48
pretrained	1e-6	Flow1s	0.46	0.73
pretrained	1e-6	Flow2s	0.57	0.36
pretrained	1e-6	Flow3s	0.56	0.37
random	1e-5	Flow1s	0.45	0.66
random	1e-5	Flow2s	0.56	0.43
random	1e-5	Flow3s	0.57	0.43
random	1e-6	Flow1s	0.44	0.84
random	1e-6	Flow2s	0.59	0.3
random	1e-6	Flow3s	0.58	0.33

Table 2: Validation results for the proposed architectures (Flow1s, Flow2s, Flows3) and the benchmark (GOTURN). P_0 and γ refer to the initialization method and base learning rate, respectively, of the convolutional parameters applied to the flow input. See Figure 1 for details of the architectures.

Final Test

This section presents the VOT2015 test results for the architectures using the settings chosen during validation. Table 3 shows the results for the full architectures, Table 4 for the lighter architectures that have fewer parameters in the fully-connected branch. In addition to average Overlap and Failures, the VOT toolkit computes *Expected Overlap* that estimates the short-term overlap between the predicted and real bounding boxes as if the tracker was not initialized after failures. The results using the VOT2016 test videos are fairly similar and thus not documented.

Beginning with the full architectures (Table 3), the Flow1s architecture (only optical flow input) is clearly the worst tracker. Considering Flow2s, Flow3s and GOTURN, Overlap and Failures are on average very similar regardless of the architecture. According to Expected Overlap, the final performance measure of VOT, GOTURN is slightly the best and Flow3s somewhat better than Flow2s. However, the differences are very small. Comparing the full and light architectures (Tables 3 and 4), decreasing the number of parameters

	Expected Overlap	Overlap	Failures
GOTURN	0.1691	0.41	2.69
Flow1s	0.0818	0.29	5.85
Flow2s	0.1557	0.41	2.47
Flow3s	0.1646	0.40	2.68

Table 3: Test results for VOT2015 using full architectures.

	Expected Overlap	Overlap	Failures
GOTURN	0.1216	0.40	3.85
Flow1s	0.0829	0.29	5.29
Flow2s	0.1348	0.40	3.10
Flow3s	0.1322	0.39	3.37

Table 4: Test results for VOT2015 using light architectures.

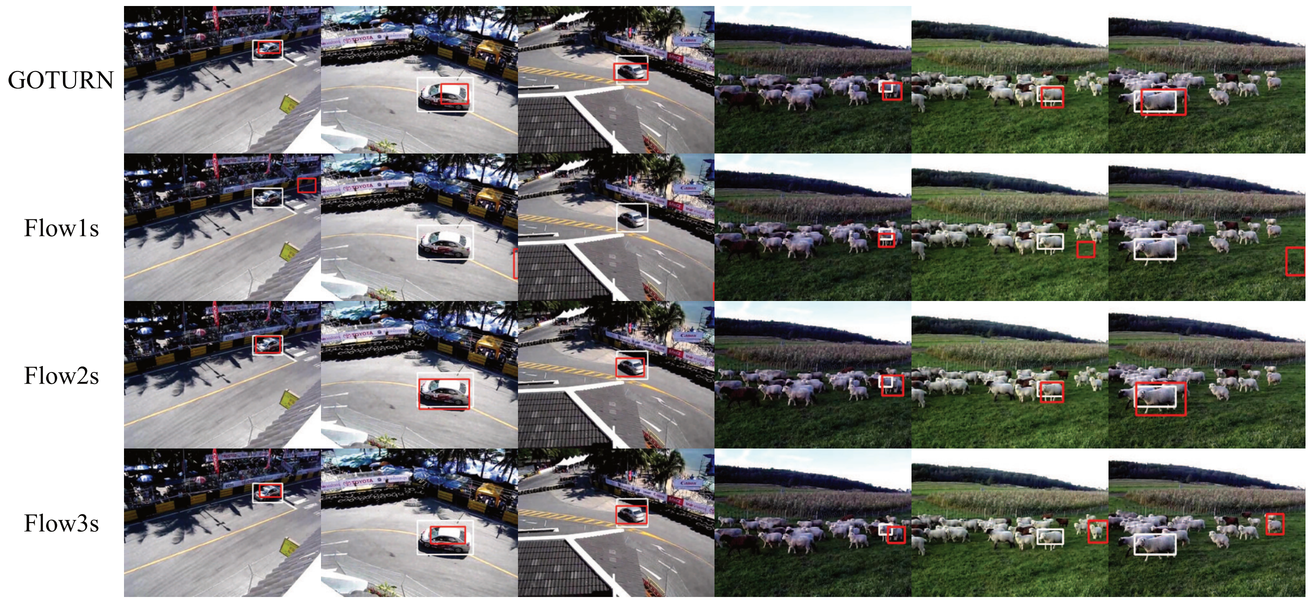

Figure 2: Examples of tracking results. Three selected frames from VOT2015 sequences *racing* and *sheep*. The rows show results for GOTURN, Flow1s, Flow2s and Flow3s, respectively. Videos play forwards from left to right. White rectangles show the ground-truth bounding boxes (slightly distorted by converting rotated boxes to axis-aligned boxes for visualization). Red rectangles show the bounding boxes estimated by the different architectures. (Best viewed in color. Color version online.)

in the fully-connected branch has practically no effect on Overlap but increases Failures. Expected Overlap decreases notably due to the increased likelihood of tracking failures.

Figure 2 shows examples of the test sequences. They were chosen randomly, mainly to visualize the dataset and typical tracking errors. In these cases, Flow1s loses track of the target quickly and is unable to recover. Since tracking is based on frames cropped around the previous location, all presented trackers face the risk of permanently losing track. The accuracy of the bounding boxes varies from frame to frame differently for each tracker, but GOTURN, Flow2s and Flow3s are often able to recover unless the location estimate is too inaccurate. In the *sheep* sequence, Flow3s starts to follow a wrong sheep that was near the original target in the beginning of the video. This shows a typical error of the trackers; they easily start to follow incorrect objects.

Discussion

Comparing the structure and performance of the architectures allows making many remarks about the role of optical flow in deep visual tracking. The original GOTURN tracker performs moderately even with a single image frame, but its performance improves when it uses both the current and previous frame (for details, see (Held, Thrun, and Savarese 2016)). This suggests that the performance of GOTURN relies on static features available in a single image and on motion features available by processing two consecutive frames. Any state information over consecutive frames can allow a network to learn motion features implicitly. The Flow1s and Flow2s architectures mostly remove this ability since the only state information they use is the cropping window centered on the previous location estimate.

Replacing GOTURN by Flow2s, i.e. using the current frame but replacing the previous frame explicitly by optical flow, does not seem to affect the tracking performance. Thus, the explicit flow input is effective in the sense that it can replace the implicit motion representations. Moreover, since Flow2s performs about equally well as GOTURN, the flow input seems to capture most of the benefits of implicit motion representations. This suggests that GOTURN learns implicitly some representation of optical flow. Since GOTURN benefits from its motion representations, optical flow is useful for tracking.

Replacing GOTURN or Flow2s by Flow1s, i.e. using explicit optical flow as the sole input, allows assessing the importance of static features GOTURN learns from single images. Flow1s does not provide satisfactory performance. Thus, the studied combination of explicit representation of optical flow (Farneback method) and tracking approach (architecture, bounding box output) is not sufficient for visual tracking. The raw images contain additional information that GOTURN and Flow2s learn to utilize. Flow1s performs worse since it cannot use static single-frame features.

Replacing GOTURN by Flow3s, i.e. complementing the current and previous frames with optical flow, does not seem to affect the tracking performance. Allowing the network to learn implicit representations of optical flow (giving it both input images) while using also the explicit flow input does not improve the performance. Thus, using either one of the representations, implicit or explicit, is enough in terms of the accuracy of the tracker. Potential benefits of using more information might be canceled out by a higher risk of overfitting or getting contradicting input information.

The notes so far show that GOTURN learns implicitly effective representations of optical flow and that the implicit representations can be replaced by explicit ones. However, the implicit representations are enough and incorporating optical flow more explicitly into the network does not improve the tracking accuracy. Comparing the performance of the full and light architectures (Tables 3 and 4) could reveal some benefits from explicit optical flow. The light architectures have a lower learning capacity due to having fewer parameters. The flow input seems more useful in the light architectures since relying more on optical flow leads to lower decrease in performance. Moving from full to light architectures, the decrease in Expected Overlap is 28% for GOTURN, 20% for Flow3s and only 13% for Flow2s. The performance of Flow1s actually improves using the light architecture. This can suggest that richer flow input simplifies learning. Although the accuracy of the tested light architectures is low, using optical flow could allow learning lighter and more efficient networks for tracking. This aspect should be studied in more detail in future research.

The main goal of this paper is to analyze the role of optical flow in deep visual tracking, not necessarily to develop highly successful trackers. Nevertheless, it is important to compare the performance to state-of-the-art trackers. The proposed architectures build on the GOTURN tracker (Held, Thrun, and Savarese 2016), and except for Flow1s they perform very similarly to GOTURN, yielding Expected Overlap of around 0.16–0.17. In terms of the VOT results, GOTURN is only an average tracker. This highlights the need for improving the proposed trackers. For instance, recent publication (Zhu et al. 2018) that also uses optical flow for tracking reaches Expected Overlap of 0.34 for VOT2015 videos, which makes it one of the best trackers. This shows that optical flow can be used to build very good trackers.

There are many ways to potentially improve the proposed trackers or to highlight the benefits of using optical flow explicitly. Several simplifications were made while adapting GOTURN to a different training dataset to properly use the VOT videos for testing. Compared to the original paper (Held, Thrun, and Savarese 2016), GOTURN performs worse when trained in our framework (Expected Overlap falls from about 0.20 to 0.17), which likely affects also the proposed architectures. It is important to find ways to fully utilize the larger dataset for training. Another option is to analyze the robustness to appearance changes (e.g. occlusion, illumination) provided in some tracking datasets. This could emphasize the effectiveness of explicit flow in some cases. Finally, this paper considers only one traditional method to estimate optical flow (Farneback method). Using more modern methods could improve performance, especially for Flow1s that relies solely on the explicit flow input.

This paper focuses on just one approach to deep visual tracking, the GOTURN architecture. Deep trackers are often notably different from each other. Methods to identify and remove the implicit representations of optical flow and to replace them with explicit representations are different for each tracking approach. We deliberately concentrate on one intuitive tracker to make the analysis and presentation relatively comprehensive and clear. Studying the role of optical flow in other deep trackers is necessary to generalize the results. It is also important to search for more natural and effective ways of using optical flow in visual tracking.

Conclusions

This paper examines the role of optical flow in deep visual tracking. The results show that the considered tracker learns implicit representations of optical flow. The representations are useful, but effective tracking requires also other information present in static image frames. The implicit representations can be replaced by giving the optical flow explicitly as an input to the network. Although the network learns sufficient representations of optical flow implicitly, incorporating explicit flow information into the network could allow constructing lighter and more efficient models for deep visual tracking.

Acknowledgments

Strategic Research Council at the Academy of Finland is acknowledged for financial support for project ”Competence Based Growth Through Integrated Disruptive Technologies of 3D Digitalization, Robotics, Geospatial Information and Image Processing/Computing - Point Cloud Ecosystem (293389 / 314312)”.

References

Bertinetto, L.; Valmadre, J.; Golodetz, S.; Miksik, O.; and Torr, P. H. S. 2016a. Staple: Complementary learners for real-time tracking. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1401–1409.

Bertinetto, L.; Valmadre, J.; Henriques, J. F.; Vedaldi, A.; and Torr, P. H. S. 2016b. Fully-convolutional siamese networks for object tracking. In Hua, G., and Jégou, H., eds., *Computer Vision – ECCV 2016 Workshops*, 850–865. Springer International Publishing.

Bolme, D. S.; Beveridge, J. R.; Draper, B. A.; and Lui, Y. M. 2010. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2544–2550.

Danelljan, M.; Khan, F. S.; Felsberg, M.; and v. d. Weijer, J. 2014. Adaptive color attributes for real-time visual tracking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1090–1097.

Danelljan, M.; Häger, G.; Khan, F. S.; and Felsberg, M. 2015. Learning spatially regularized correlation filters for visual tracking. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 4310–4318.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazirbas, C.; Golkov, V.; v. d. Smagt, P.; Cremers, D.; and Brox, T. 2015. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2758–2766.

Farnebäck, G. 2003. Two-frame motion estimation based on polynomial expansion. In Bigun, J., and Gustavsson, T., eds., *Image Analysis – 13th Scandinavian Conference*, 363–370. Springer Berlin Heidelberg.

Galoogahi, H. K.; Fagg, A.; and Lucey, S. 2017. Learning background-aware correlation filters for visual tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 1144–1152.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 580–587.

Guo, Q.; Feng, W.; Zhou, C.; Huang, R.; Wan, L.; and Wang, S. 2017. Learning dynamic siamese network for visual object tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 1781–1789.

Held, D.; Thrun, S.; and Savarese, S. 2016. Learning to track at 100 fps with deep regression networks. In Leibe, B.; Matas, J.; Sebe, N.; and Welling, M., eds., *Computer Vision – ECCV 2016*, 749–765. Springer International Publishing.

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

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia (MM)*, 675–678. New York, NY, USA: ACM.

Kristan, M.; Matas, J.; Leonardis, A.; Felsberg, M.; Cehovin, L.; Fernandez, G.; Vojir, T.; Hager, G.; Nebehay, G.; Pflugfelder, R.; Gupta, A.; Bibi, A.; Lukezic, A.; Garcia-Martin, A.; Saffari, A.; Petrosino, A.; and Montero, A. S. 2015. The visual object tracking vot2015 challenge results. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 564–586.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems (NIPS) 25*. Curran Associates, Inc. 1097–1105.

Li, B.; Yan, J.; Wu, W.; Zhu, Z.; and Hu, X. 2018. High performance visual tracking with siamese region proposal network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8971–8980.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440.

Ma, C.; Huang, J. B.; Yang, X.; and Yang, M. H. 2015. Hierarchical convolutional features for visual tracking. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 3074–3082.

Nam, H., and Han, B. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4293–4302.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NIPS) 28*. Curran Associates, Inc. 91–99.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* 115(3):211–252.

Simonyan, K., and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems (NIPS) 27*. Curran Associates, Inc. 568–576.

Smeulders, A. W. M.; Chu, D. M.; Cucchiara, R.; Calderara, S.; Dehghan, A.; and Shah, M. 2014. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(7):1442–1468.

Song, Y.; Ma, C.; Gong, L.; Zhang, J.; Lau, R. W. H.; and Yang, M. 2017. Crest: Convolutional residual learning for visual tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2574–2583.

Song, Y.; Ma, C.; Wu, X.; Gong, L.; Bao, L.; Zuo, W.; Shen, C.; Lau, R. W. H.; and Yang, M. 2018. Vital: Visual tracking via adversarial learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8990–8999.

Tao, R.; Gavves, E.; and Smeulders, A. W. M. 2016. Siamese instance search for tracking. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1420–1429.

Valmadre, J.; Bertinetto, L.; Henriques, J.; Vedaldi, A.; and Torr, P. H. S. 2017. End-to-end representation learning for correlation filter based tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5000–5008.

Wang, L.; Ouyang, W.; Wang, X.; and Lu, H. 2015. Visual tracking with fully convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 3119–3127.

Wu, Y.; Lim, J.; and Yang, M. H. 2015. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(9):1834–1848.

Zhu, Z.; Huang, G.; Zou, W.; Du, D.; and Huang, C. 2017. Uct: Learning unified convolutional networks for real-time visual tracking. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 1973–1982.

Zhu, Z.; Wu, W.; Zou, W.; and Yan, J. 2018. End-to-end flow correlation tracking with spatial-temporal attention. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 548–557.