Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2016

Visual Tracking Using Deep Motion Features

Susanna Gladh

Master of Science Thesis in Electrical Engineering

Visual Tracking Using
Deep Motion Features

Susanna Gladh

LiTH-ISY-EX--16/5005--SE

Supervisor: **Martin Danelljan**
ISY, Linköpings universitet

Examiner: **Fahad Khan**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Sammanfattning

Generisk visuell följning är ett utmanande problem inom datorseende, och innebär att positionen för ett objekt estimeras i en sekvens av bilder, med endast startpositionen given. Programmet måste därför kunna lära sig ett objekt och beskriva det genom dess visuella egenskaper.

Vanliga deskriptorer använder sig av statisk information i bilden. I det här examensarbetet undersöks möjligheten för användning av optiskt flöde och djup inärning för objektinformation. Optiska flödet räknas ut via två på rad följande bilder, och beskriver därmed dynamisk information i bilden. Detta visar sig i evalueringarna som utförts vara ett utmärkt komplement till standarddeskriptorerna. Vidare har metoderna PCA och PLS, för dimensionalitetsreduktion av deskriptorerna, utvärderats i det framtagna programmet. Resultaten visar att bägge metoderna förbättrar programmets förmåga och hastighet ungefär lika mycket, men att PLS faktiskt gav något bättre resultat jämfört med populära PCA.

Det slutgiltiga implementerade programmet testades på tre utmanande dataset och uppnådde bäst resultat vid jämförelse med andra toppmoderna program som utför visuell följning.

Abstract

Generic visual tracking is a challenging computer vision problem, where the position of a specified target is estimated through a sequence of frames. The only given information is the initial location of the target. Therefore, the tracker has to adapt and learn any kind of object, which it describes through visual features used to differentiate target from background.

Standard appearance features only capture momentary visual information. This master's thesis investigates the use of deep features extracted through optical flow images processed in a deep convolutional network. The optical flow is calculated using two consecutive images, and thereby captures the dynamic nature of the scene. Results show that this information is complementary to the standard appearance features, and improves performance of the tracker.

Deep features are typically very high dimensional. Employing dimensionality reduction can increase both the efficiency and performance of the tracker. As a second aim in this thesis, PCA and PLS were evaluated and compared. The evaluations show that the two methods are almost equal in performance, with PLS actually receiving slightly better score than the popular PCA.

The final proposed tracker was evaluated on three challenging datasets, and was shown to outperform other state-of-the-art trackers.

Acknowledgments

I begin with a large thank you to my supervisor Martin Danelljan and examiner Fahad Khan, who helped and inspired me through the thesis work.

I thank Michael Felsberg for valuable comments and opinions. Thank you to Guilia Meneghetti for helping me with setting up my work station and providing technical support, and for the uplifting conversations about our cats and lives in general. Also, I thank my family and my cousin Emma, who all tried to understand and show interest when I updated them about my progress with the thesis.

Finally, thank you to my boyfriend, who always listen, believe in and support me.

Linköping, October 2016
Susanna Gladh

Contents

Notation

Sets

Notation	Meaning
$\mathbb{R}$	The set of real numbers

Abbreviations

Notation	Meaning
HOG	Histograms of Oriented Gradients
DFT	Discrete Fourier Transform
DCF	Discriminative Correlation Filter
SRDCF	Spatially Regularized DCF
CNN	Convolutional Neural Network
PLS	Partial Least Squares
PCA	Principal Component Analysis
OP	Overlap Precision
AUC	Area Under Curve

Functions and Operators

Notation	Meaning
$\lvert\cdot\rvert$	Absolute value
$\lVert\cdot\rVert$	L^2-norm
$*$	Circular convolution
∇	Gradient
$\mathscr{F}$	The Discrete Fourier Transform
$\mathscr{F}^{-1}$	The Inverse Discrete Fourier Transform

1 Introduction

Visual tracking is a computer vision problem, where the task is to estimate the location of a specific target in a sequence of images, i.e. a video. This is an important and challenging assignment, with many real-world applications, such as robotics, action recognition, or surveillance. The tracker must be able to handle complicated situations, including fast motion, occlusion, and different appearance changes of the target, for example rotations or deformation.

The main aim of this thesis is to explore visual tracking using deep motion features. The other aim is to evaluate two forms of dimensionality reduction techniques on the extracted deep features, through observation of the tracking performance during different settings.

In this chapter follows an introduction with some general information about generic visual tracking. After this familiarization with tracking follows a motivation in section 1.2, which aims to support the aim and problem formulation of the thesis described in section 1.3. The scope is explained in section 1.4, followed by a summarization of the results and contributions in section 1.5. Lastly, an outline of the remainder of this thesis is provided in section 1.6.

1.1 An Introduction to Generic Visual Tracking

The intention with this section is to give a brief overview of generic visual tracking in order to give the reader a starting point for the rest of the thesis, and a better understanding of the motivation and aims. The provided information is meant to cover the scope of this thesis. For a more complete survey of tracking and different tracking methods, see [32].

In visual tracking, the computer vision task is to estimate the position of an object through a sequence of frames. An example of this can be seen in figure 1.1, where a panda is

Figure 1.1: *A frame capture from the sequence Panda (left image), in which a surveillance camera monitors a panda walking about its enclosed living area. The same frame is seen as output from a tracker with the task of estimating the panda's trajectory (right image). The red box represents the estimated location given by the tracker.*

tracked in a video sequence named *Panda*, captured by a surveillance system monitoring the panda. During tracking, the estimated location of, in this case, the panda is marked by a rectangular shaped bounding box (right image in figure 1.1), which should fit the object's size, shape and spatial location in the image.

The object to be tracked appertains to a category of objects, e.g. humans, dogs, cars (or pandas), and the tracker is therefore dependent on having an initial model of the object, which also can be updated after each processed frame during the tracking process. This model can be constructed using information from other objects in the same category as the object to be tracked. A highly important part of the model is visual representation, or the choice of *visual features*. By extracting object-specific features, a descriptor is constructed, with the task of storing the data that best describe the wanted object for future comparison. The most common choices of feature representations are edges, pixel intensity, or color. Section 2.4 will provide more insight into visual features.

In *generic tracking*, the task gets more challenging, as only the initial location of the object is provided. Taking figure 1.1 as an example, this means that the red box is provided in the first frame, nothing else, and then the tracker is left on its own to learn the object features. In other words, the tracker cannot use an initial model of the object to be tracked, since it does not know what category of objects it belongs to. Given this starting position of the object and/or the initial model (depending on the case of regular or generic visual tracking), the next step is to estimate the object position in the following frame. One way of doing this is to apply some kind of machine learning, and train a classifier to differentiate *target* from *background*. The result will provide the most probable target location in the image. This approach is called *tracking by detection*, and it is the method employed by most state-of-the-art trackers.

1.2 Motivation

Visual tracking is a highly difficult problem with a large range of applications. Most existing tracking approaches employ hand-crafted appearance features, such as HOG (histogram of oriented gradients) and Color Names [25, 8, 10]. However, deep appearance features extracted by feeding an RGB image to a convolutional neural network (CNN) have recently been successfully applied for tracking purposes [11, 36].

Despite their success, deep appearance features still only capture *static* appearance information, and tracking methods solely based on such features struggle in scenarios with, for example, fast motion and rotations, background distractors with similar appearance as target object, and distant or small objects (examples of such scenarios can be seen in figures 4.5 and 4.11). In cases as these, high-level *motion features* could provide rich complementary information that can disambiguate the target.

While appearance features only encode momentary information from a single frame, deep motion features integrate information from the consecutive *pair* of frames used for estimating the *optical flow*. Deep motion features can therefore capture the dynamic nature of a scene, providing complementary information to the appearance features. This raises motivation to investigate the fusion of standard appearance features with deep motion features for visual tracking.

The robustness and accuracy of a tracker are the basic and most important measurements how successful it is. However, efficiency is also of great importance, especially in the case of real-time tracking. Many state-of-the-art trackers show good performance in robustness and accuracy, while overlooking the efficiency part. It would be interesting to look into ways of improving the efficiency of the proposed tracker. Dimensionality reduction of the high-dimensional features would reduce the computational workload and thereby decrease the time taken for each frame to be processed by the tracker. Furthermore, such techniques are also known to provide better classification scores, by removing redundant information [38]. Principal Component Analysis (PCA) is the most commonly used method, but recent work [47] in computer vision has used Partial Least Squares (PLS) with good results. Perhaps PLS could be an option in tracking purposes as well.

1.3 Aim

The first aim of this master thesis is to investigate the use and impact of deep motion features in a tracking-by-detection framework. The other part of this thesis aims to evaluate the efficiency gain and performance using the two different dimensionality reduction techniques PCA and PLS, and compare these methods to see if PLS is a good alternative to PCA.

The approach of the first part can be formulated through the following questions:

- How do deep motion features affect performance in a tracking-by-detection framework?
- Are deep motion and appearance features complementary?

For the second part, the approach will be to investigate the following:

- What are the pros and cons of the selected dimensionality reduction techniques when compared to each other?
- Which of the two methods is more preferable?

1.4 Scope

Since the main aim is to investigate the *impact* of adding motion features to a tracking-by-detection paradigm, the choice was made to use only pre-trained CNNs. Therefore, the theory on CNNs is kept at a somewhat limited level. Also, obtaining real-time speed of the tracker has not been a priority.

1.5 Contributions

This thesis investigates the impact of deep motion features for visual tracking and evaluates the use of two dimensionality reduction techniques.

To extract deep motion features, a deep optical flow network pre-trained for action recognition was used. The thesis includes investigating the fusion of *hand-crafted* and *deep* appearance features with deep motion features, in a state-of-the-art discriminative correlation filter (DCF) based tracking-by-detection framework [10]. To show the impact of motion features, evaluations are performed with fusions of different feature combinations. Furthermore, an evaluation of two dimensionality reduction methods is performed, comparing potential gains in efficiency and accuracy. Finally, to validate the performance of the resulting implemented tracker, extensive experiments are performed on three challenging evaluation datasets, and the results are compared with current state-of-the-art methods.

The results show that the fusion of deep and hand-crafted appearance features, with deep motion features significantly improves the baseline method, employing only appearance features. Also, the addition of dimensionality reduction improves the result even further, and provides some efficiency gain. The proposed tracker was shown to outperform the other methods in the comparison with state-of-the-art trackers.

Part of this thesis has been published [22] at the International Conference on Pattern Recognition (ICPR) 2016. The paper[1] was awarded *Best Scientific Paper Prize* for track in Computer Vision and Robot Vision.

1.6 Thesis outline

Following this introduction, chapter 2 combines some theory and related work about discriminative tracking, deep convolutional neural networks, visual features and dimensionality reduction. It also includes information about the tracker employed as the baseline framework in this thesis.

[1] The paper is available at `https://arxiv.org/abs/1612.06615`

Chapter 3 describes the thesis methodology, starting with the extraction and fusion of the different kinds of features employed, followed by the implementation method of the dimensionality reduction, and finally how multiple feature types were integrated in the SRDCF framework.

All of the evaluations, including a discussion after each evaluation, are gathered in chapter 4, followed by some final conclusions and thoughts on future work in chapter 5.

2

Theory and Related Work

A short introduction on generic visual tracking was provided in section 1.1. The starting point of this thesis was the SRDCF (short for Spatially Regularized Correlation Filter) tracking framework, in which a discriminative correlation filter is trained and applied to the visual feature map in order to estimate the target location. Section 2.2 explains further how the SRDCF tracker works.

Before going into detail about the SRDCF, some theory embedded with some related work on discriminative trackers is explained in section 2.1. A brief insight on convolutional neural networks is explained in section 2.3, followed by information about the visual feature representations employed in the thesis in section 2.4. Lastly, section 2.5 explains a bit of theory behind the dimensionality reduction techniques evaluated during this thesis work.

2.1 Discriminative Trackers

There are multiple different tracking techniques, and the approaches are traditionally categorized into generative [15, 20, 28] and discriminative [23, 24, 58, 36, 10] methods. The first mentioned category applies generative learning methods in the construction of the appearance model, which estimates a joint probability distribution of the features and their labels. Discriminative methods on the other hand, aim to, given the features, model the probability distribution of their labels. The discriminative methods typically train a classifier or regressor, with the purpose of differentiating target from background.

Discriminative methods are often also termed *tracking-by-detection* approaches, since they apply a discriminatively trained classifier or regressor, which is trained online (during tracking), by extracting and labeling samples from the video frames. These training samples, or features, are often represented by for example raw image patches [2, 26],

image histograms and Haar features [23], color [9, 58], or shape features [8, 25].

One group of discriminative tracking methods, which is of importance for this thesis, uses correlation filters, and will be introduced in the following section.

2.1.1 DCF based trackers

Among the discriminative, or tracking-by-detection approaches, the Discriminative Correlation Filter (DCF) based trackers [2, 8, 10, 25, 36] have recently demonstrated excellent performance on standard tracking benchmarks [56, 31]. The key for their success is the ability to efficiently utilize limited data by including all shifts of local training samples in the learning.

DCF-based methods train a least-squares regressor by minimizing the L^2-error between the responses (classified features) and the labels. The problem can then efficiently be handled by Fourier transforming the least squares problem, and solving the resulting linear equations. The result is a Discrete Fourier Transformed (DFT) correlation (or convolution) filter. The classification is made by circular correlation of the filter and the extracted feature map. This too can be handled in the Fourier domain, which means that the classification can be made by point-wise multiplication of the Fourier coefficients of the feature map and filter. The result gives a prediction of the target confidence scores, which can be seen as a confidence map over where the target is located.

The MOSSE tracker [2] first considered training a single feature channel (single-channel) correlation filter based on gray-scale image samples of target and background appearance. A remarkable improvement is achieved by extending the MOSSE filter to multi-channel features. This can be performed by either optimizing the exact filter for offline learning applications [18, 1] or using approximative update schemes for online tracking [8, 9, 25].

As mentioned, the success of DCF methods is dependent on the use of circular correlation,

(a) Original image. (b) Periodicity in correlation filters.

Figure 2.1: *A frame capture from the sequence Soccer (a) with target marked with a green rectangle, and a visualization of the periodic assumption (b) used by DCF trackers. The image is from the SRDCF paper [10].*

which enables fast calculations and increased amount of training data. However, circular correlation assumes periodic extensions of the training sample patch, see figure 2.1, which will introduce boundary effects at the periodic edges, which in turn affect the training and detection. Because of this, a restricted training and search region size is required. This problem has recently been addressed by performing a constraint optimization [19, 17]. These approaches are however not practical when using multiple feature channels and online learning. Instead, Danelljan et al [10] proposed the Spatially Regularized DCF (SRDCF), where a spatial penalty function (see figure 2.2) is added as a regularizer in the learning. This can be seen as a weight function that penalizes samples in the background (outside the target region) of the sample, thereby alleviating the periodic effects, allowing a larger sample size, and increasing the training data, while still maintaining the effective size of the filter. The filter is also optimized in the Fourier domain using Gauss-Seidel iterations. The approach leads to a remarkable gain in performance compared to previous approaches. The weight function is shown in figure 2.2, and a comparison of the standard DCF and SRDCF is displayed in figure 2.3.

While the original SRDCF employs hand-crafted appearance features (HOG), the Deep-SRDCF [11] investigates the use of convolutional features from a deep RGB network in the SRDCF tracker (see section 2.4 for information about visual features). In this thesis, the proposed tracking framework is also based on the SRDCF framework. For this reason, the following section will explain the SRDCF further.

2.2 The SRDCF tracker

As mentioned in the previous section, the Spatially Regularized Discrete Correlation Filter (SRDCF) tracker [10] has recently been successfully used for integrating single-layer deep features [11] (see section 2.4.2 for information about deep features). This section will introduce the basics of the SRDCF framework, which is employed as a baseline tracker in this thesis. For a more detailed description, see [10].

Utilizing the FFT provides a desirable efficiency gain during training and detection. How-

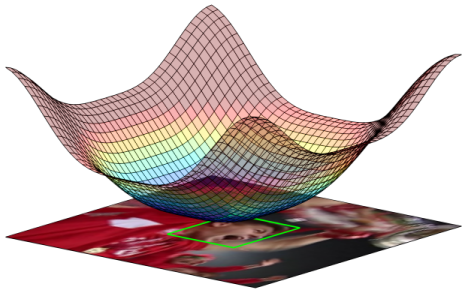

Figure 2.2: *A graphic interpretation of the regularization weight function w employed during training in the SRDCF tracker. The background data is penalized through the larger filter coefficients. The image is from the SRDCF paper [10].*

ever, as mentioned, the periodic assumption also leads to unwanted boundary effects, as seen in figure 2.1, and a restricted size of the image region used for training the model and searching for the target. The purpose behind the development of the SRDCF was to address these problems, which was achieved by introducing a spatial regularization weight in the learning formulation.

The framework includes learning of a discriminative convolution filter with training samples $\{(x_k, y_k)\}_{k=1}^{t}$. The multi-channel feature map x_k contains d dimensions (channels) and has spatial size $M \times N$. The specific feature channel l of x_k is denoted x_k^l, and is extracted from an image patch containing both target and large amounts of background information. y_k is the corresponding label at the same spatial region as x_k, and consists of the desired $M \times N$ confidence score function. In other words, $y_k(m, n) \in \mathbb{R}$ is the desired classification confidence at the spatial location (m, n) in the feature map x_k. A Gaussian function is used to determine these desired scores y_k.

The trained convolution filter will consist of one $M \times N$ filter f^l per feature dimension l, visualized in figure 2.3. The target confidence scores $S_f(x)$ for an $M \times N$ feature map x are computed in the following manner:

$$S_f(x) = \sum_{l=1}^{d} x^l * f^l \,, \tag{2.1}$$

where $*$ denotes circular convolution. Calculating $S_f(x)$ will apply the linear classifier f at all locations in the feature map x. The circular convolution will implicitly extend the boundaries of x. To learn f, the squared error $\epsilon(f)$ between the confidence scores $S_f(x_k)$ and the corresponding desired scores y_k is minimized as such,

$$\varepsilon(f) = \sum_{k=1}^{t} \alpha_k \left\| S_f(x_k) - y_k \right\|^2 + \sum_{l=1}^{d} \left\| w \cdot f^l \right\|^2 , \tag{2.2}$$

where $\cdot$ denotes point-wise multiplication. Here, the sample weights α_k are exponentially decreasing and will determine the impact of the training samples, and the spatial regular-

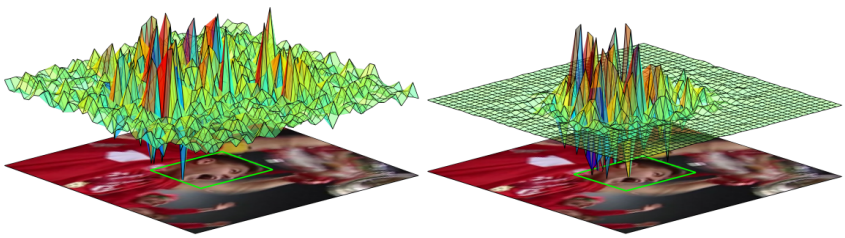

Figure 2.3: *Comparison of the standard DCF (left) and the SRDCF (right). While the DCF has large filter coefficients residing in the background, the SRDCF emphasizes the information near the target region. This is due to the weight function w penalizing filter coefficients f^l further away from the target region. The image is from the SRDCF paper [10].*

ization term is determined through the penalty weight function w, which is visualized in figure 2.2. As seen in the figure, the weights will penalize large filter coefficients in image regions outside the target region, which correspond to background information. This sets the effective filter size to the target size in the feature map, while reducing the impact of background features. Due to this, larger training samples x_k can be used, which leads to a large gain in negative training data, without increasing the effective filter size.

Since w has a smooth appearance, the Fourier spectrum will be sparse (most elements are zero), which enables the use of sparse solvers. To train the filter, $\epsilon(f)$ (2.2) is minimized in the Fourier domain using iterative sparse solvers.

2.3 Convolutional Neural Networks

A deep Convolutional Neural Network (CNN) applies a sequence of operations to a raw image patch, where each operation is referred to as a layer of the network [39]. The operations are normally convolution, Rectified Linear Unit (ReLU), local response normalization, and pooling operations.

An artificial neuron takes several inputs, which can be weighted according to importance, and produces a single output. If all input neurons are connected to each neuron in the current layer, the layer is referred to as a *fully connected* (FC) layer. This is typically the case for the final layers in a CNN, which also include an output classification layer. Fully connected layers does not take the spatial structure of the image into account, as pixels far apart and close together are handled on the same footing, which is why the use of such layers is not optimal for image classification. [39]

The convolution layer consists of a grid of neurons, and takes a grid of neuron activations (or the input pixels) as input. The operation is a convolution of the neurons and the input, i.e. an image convolution, with weights specifying the convolution filter. Each neuron in the convolutional layer will be connected to a small region of the input, known as the *local receptive field* of the neuron. A visualization of this is seen in figure 2.4. The distance between the center of each respective local field is called the *stride length*. [39]

After a convolutional layer, there may be a pooling layer, which condenses the output from the previous layer by taking small blocks, and sub-sampling each block to a single output. This can be done through for example averaging, or taking the maximum activation value in the block, where the last mentioned is known as max-pooling. Another layer that follows a convolutional layer is a ReLU-operation layer. This layer employs the rectifier as activation function, which is defined as $f(x) = max(0, x)$, where x is the input to a neuron. Sometimes there is also a local response normalization layer, which, as the name implies, performs normalization over local regions around the neurons. This will boost neuron activations that are large compared to neighboring neurons, and suppress areas where neurons have more uniform responses, hence providing better contrast. [39]

The parameters of the network are chosen trough training using large amounts of labeled images, such as the ImageNet dataset [46], which contains about 15 million images in 22,000 different categories.

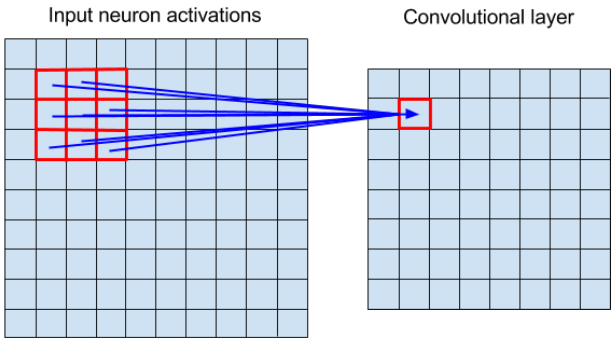

Figure 2.4: *Visualization of a convolutional layer (right) and the local receptive field (red 3-by-3 grid) of one of the neurons in the input layer (left).*

2.4 Visual Features

The visual feature representation is a very important component in the tracking framework. These features describe characteristics in the image, and the aim is to capture object specific information, so that the target can be found in the next frame. In generic tracking, where the object category is unknown and can vary between each sequence given to the tracker, it is particularly important to find different kinds of features that describe the current object. When talking about visual features, the terms high- and low-level information are often used. In this context, low-level information describes rather specific image characteristics, whereas high-level is a more abstract description.

2.4.1 Hand-crafted Features

Hand-crafted features, sometimes also referred to as shallow features, are typically used to capture low-level information. This includes for example shape, color or texture. A popular feature representation is the Histograms of Oriented Gradients (HOG) [7]. These features mainly capture information regarding the shape by calculating histograms of gradient directions in a spatial grid of cells, which is normalized with respect to nearby cells in order to add invariance.

The Color Names (CN) descriptor [54] applies a pre-learned mapping from RGB to the probabilities of 11 linguistic color names. The mapping results in an image, where the pixel values represent the probability of corresponding pixels in the original image having the same color.

Hand-crafted Features in Computer Vision

HOG features have been used for both visual tracking [8, 10, 25] and object detection [16]. Feature representations based on color, such as color transformations [41, 58] or color histograms [42], have also been commonly employed for tracking. CN features have discriminative power as well as compactness, and have been successfully applied in

tracking [9].

2.4.2 Deep Features

The output from each layer in a CNN (see section 2.3), i.e. the activations of the neurons in the layer, is also referred to as a feature map. This can be seen as the kind of spatial structure, or pattern, that will cause activation during detection. Features could for example be edges or shapes of different kinds. Features from a CNN are also often called *deep features*, which is the term that will be used further on in this thesis.

The convolutional layer to the right in figure 2.4 has dimensions $N \times M \times d$, where d in this specific example is equal to 1. This produces a single-channel output of activations referred to as a (single-channel) feature map. Generally, a layer in a CNN in practice has d greater than 1, resulting in a multi-dimensional feature map. This means that the layer is capable of detecting more than one feature type per layer, e.g. sharp edges and different gradients etc. These multi-dimensional feature maps are also referred to as *multi-channel* feature maps. Examples of deep features can be seen in image 2.5, from both a shallow and deep layer in a CNN (first and second sub-rows respectively). In the image, the first row is from a CNN that is trained to handle RGB images. In this case, the network is referred to as an *appearance network* and the features as *deep appearance features*. The second row presents activations from a CNN that handles *optical flow* images (see section 2.4.3), which are referred to as (deep) *motion features* from a *motion network*.

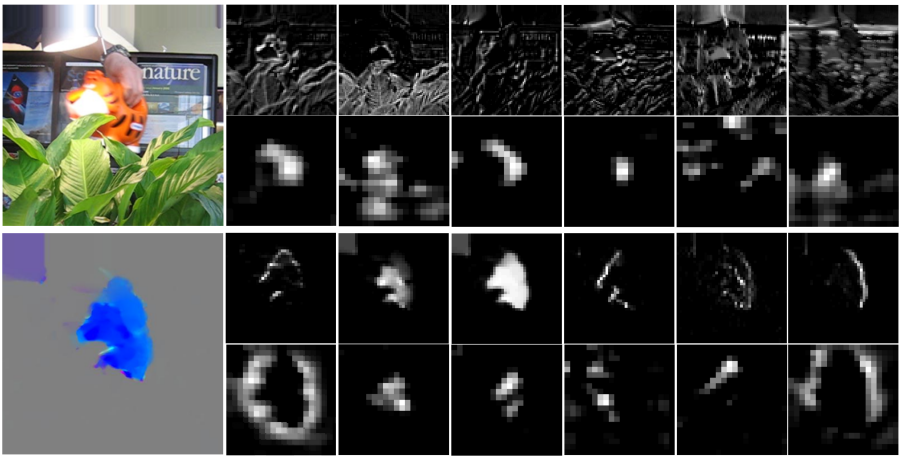

Figure 2.5: *Visualization of the deep features with highest energy (the channels with largest average activation values) from a shallow and deep CNN layer in the appearance (first and second sub-row of the top row) and motion network (first and second sub-row of the bottom row). The appearance features are extracted from the raw RGB image (top left) from the sequence Tiger2, and motion features from the corresponding optical flow image (bottom left). For both CNNs, the shallow and deep layer activations can be seen in the corresponding first and second sub-rows respectively.*

The deep features are discriminative and contain high-level visual information, while still preserving spatial structure. The features from more shallow layers encode low-level information at high spatial resolution, while the deep layer feature maps contain high-level information at a coarse resolution.

Deep Features in Computer Vision

It has been shown that features from deep convolutional layers from networks that are pre-trained for a particular vision problem, such as image classification, are generic [44]. Deep features are therefore also suitable for use other computer vision tasks, including visual tracking. As mentioned in section 2.3, the FC layers includes a classification layer, which is why most works apply the activations from the FC layer [50, 40]. In recent studies [6, 35], deep features have shown promising results for image classification. It has also been shown that deep features can be successfully applied in DCF based trackers [11, 36]. The multi-channel feature maps can be directly integrated into the SRDCF framework (see section 2.2).

2.4.3 Deep Motion Features

Deep motion features are extracted by the use of *optical flow* images. Using two consecutive images in a video sequence, the optical flow can be calculated, which describe the motion energy between the images. This is performed by estimating the displacement of pixels between the images, and represent the optical flow as the direction and distance of the estimated pixel shifts.

Optical Flow

The optical flow can be aggregated in the x-, y- direction, and together with the flow magnitude a three channel image can be constructed with pixel values (v_x, v_y, m), which can be seen as a motion vector field. For example, if an object is moving in a scene with no movement in the background, the resulting optical flow image will show the background in gray, which indicates no movement (or no optical flow), and the object will be shown in different colors depending on direction. Each pixel in the optical flow image can be interpreted as *"which direction are we moving (= color), and how fast (= intensity)?"*. This is seen clearly in the optical flow images shown in figures 2.5 and 2.6, where the tiger and panda are moving, and the background is not. In the latter case, the panda is moving in opposite direction in the first compared to the second frame shown. This will cause the color to change from pink, which represents moving in positive x-direction, to blue, representing the negative x-direction.

There are of course different approaches to estimate the optical flow. The method described below is by Brox et al. [3], and is the approach applied in this thesis. Having a gray scale image $I(x, y, t)$ at time t, the desired displacement vector between this image and one at time $t + 1$ is defined as $\mathbf{d} := (u, v, 1)^\top$, which, through the assumption that the displaced pixel value is not changed by the transition, fulfills the following [3]

$$I(x, y, t) = I(x + u, y + v, t + 1)\,. \tag{2.3}$$

Let $\mathbf{x} := (x, y, t)^\top$. An energy function that penalizes deviations from equation (2.3) can

be formed as follows [3]

$$E(u,v) = \int |I(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})|^2 + \gamma|\nabla I(\mathbf{x}+\mathbf{d}) - \nabla I(\mathbf{x})|^2 \, \mathbf{dx} \,, \tag{2.4}$$

where γ is a weight, and ∇ is the gradient. Using (2.4) as is would cause outliers to have too much impact. Therefore, a concave function $\xi(s^2)$ is applied, giving [3]

$$E(u,v) = \int \xi(|I(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})|^2 + \gamma|\nabla I(\mathbf{x}+\mathbf{d}) - \nabla I(\mathbf{x})|^2) \, \mathbf{dx} \,. \tag{2.5}$$

The expression for $\xi(s^2)$ used by [3] is $\xi(s^2) = \sqrt{s^2+\epsilon}$, where ϵ is a small positive constant that keeps $\xi(s)$ convex, which is convenient when minimizing (2.5). Lastly, another energy function controlling piecewise smoothness of the displacement field is constructed, which penalize the total variation of the field. This term is defined as [3]

$$E_s(u,v) = \int \xi(|\nabla u|^2 + |\nabla v|^2) \, \mathbf{dx} \,. \tag{2.6}$$

Figure 2.6: *Two frames from sequence Panda and corresponding optical flow images. In the first frame, the panda is moving from left to right, causing the motion energy output (optical flow image) between this frame and the one before to be represented as a pink color. Later in the sequence, the panda is moving from right to left (second image), causing the optical flow to be displayed in blue. The background is static, and will not affect the optical flow, hence the gray color.*

The total energy model can now be descried as [3]

$$E_{tot}(u, v) = E + \alpha E_s \ , \tag{2.7}$$

where $\alpha > 0$ is a regularization parameter. Now, the functions u and v that minimize (2.7) can be found using Euler-Lagrange equations and numerical approximations. Further information is provided by [3].

Deep Features Using Optical Flow

By calculating optical flow images from a dataset with large amounts of labeled videos, such as the UCF101 dataset [53], a CNN can be trained using these images as input. While appearance networks describe static information in images, deep motion networks are able to capture high-level information about the dynamic nature, i.e. motion, in the scene. Hence, features extracted using such networks can be referred to as *deep motion features*.

Deep Motion Features in Computer Vision

Recent studies [19, 5, 51, 21] have investigated the use of motion features for action recognition. Gkioxari and Malik [19] propose to use static and kinematic cues by training correlation filters for action localization in video sequences. The authors of [5] use a combination of deep appearance and motion features for action recognition. The features are extracted for different body parts, and the descriptors are then normalized and concatenated over the parts, and finally concatenated into a single descriptor with both motion and appearance features. Simonyan and Zisserman [51] have proposed an architecture that incorporates spatial and temporal networks. The network is trained on multi-frame optical flow, which results in multiple optical flow images for each frame, corresponding to flow in different directions.

Whereas the benefits of motion features have been utilized in action recognition, existing tracking methods [11, 36] are limited to using only deep appearance features from RGB images. As mentioned in section 1.3, the main aim of this thesis is to explore a combination of appearance features with deep motion for visual tracking. See section 3.1 for information about extraction of deep features in this thesis work.

2.5 Dimensionality Reduction

High dimensional data is inconvenient for several reasons. Firstly, calculations get increasingly computationally heavy with larger number of dimensions. Further more, the data will get more difficult to interpret, not just for humans but also in computer vision applications such as classification. When reducing the dimensions, redundant information and noise can be removed, which can improve the performance. [38]

The two methods for dimensionality reduction applied in this thesis are Principal Component Analysis (PCA) and Partial Least Squares (PLS), and the following sections will provide some insight to how they work. PCA is the most popular method of choice for dimensionality reduction because of its linear properties, which enable the use of a projection matrix for efficient calculations. PLS can also be used for linear regression. The

main difference from PCA is that it also is able to account for information about the class labels, which has shown to be useful in computer vision tasks, such as human detection [47].

2.5.1 PCA

As the name suggests, PCA uses information provided by the principal components to receive the directions with largest data variations. The information can then be used to create a new basis for the data. This is illustrated in figure 2.7, where the basis of the data is transformed to a new one using PCA. In case of high-dimensional data, some axes contain almost no variations, and can be removed with little to no effect on the information about the data. [38]

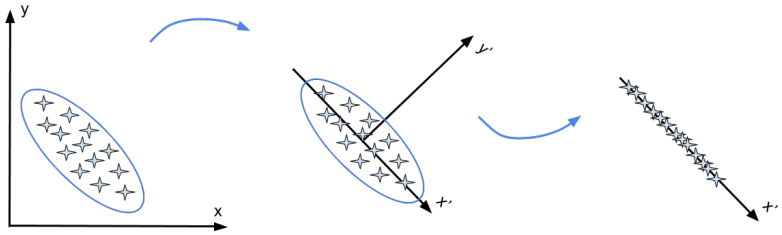

Figure 2.7: *Illustration of the principles of PCA on a two-dimensional dataset. The second set of coordinate axes is a rotation and translation of the first set, and would be found using PCA with the data (stars). The x'-axis contain very little variations compared to the y' axes, and can be removed. This leaves only the y'-axis, with the data projected onto it, and one dimension has been removed without affecting the data too much.*

Let the obtained data be stored in a $M \times N$ matrix **X'**, where M is the number of data points and N is the length of the points. In order to transform the data, it is first necessary to translate **X'** to the origin by subtracting the mean from each column, resulting in the matrix **X**. Now, the task is to find a rotation of **X**, denoted $\mathbf{P}^\top$, such that $\mathbf{Y} = \mathbf{P}^\top\mathbf{X}$, where the covariance of **Y** is diagonal [38]:

$$cov(\mathbf{y}) = cov(\mathbf{P}^\top\mathbf{x}) = \begin{pmatrix} \mathbf{p}_1^\top\mathbf{x} & 0 & ... & ... & 0 \\ 0 & \mathbf{p}_2^\top\mathbf{x} & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & ... & 0 & \mathbf{p}_M^\top\mathbf{x} \end{pmatrix}, \tag{2.8}$$

where **x** and **y** are samples of **X** and **Y**. The rows of **P** that fulfill (2.8) are called the *principal components* of **X**. The full derivation of how to find **P** will not be derived here, instead the interested reader is referred to [38] or [49]. As it turns out, the *eigenvectors* **v** of the covariance of **X** correspond to the principal components desired. The eigenvectors are orthogonal, meaning that, as an example in the two-dimensional case, one eigenvector will point in the direction with largest data variation, and the other will be perpendicular to the first one. The size of the corresponding *eigenvalues* λ are determined by the amount of data variation in the eigenvectors direction.

To get $\mathbf{P}$, we start by computing the covariance of $\mathbf{X}$ as $\mathbf{C} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$. Then, the eigenvectors $\mathbf{v}$ and eigenvalues λ of $\mathbf{C}$ are calculated so $\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$, where the columns of $\mathbf{V}$ are the eigenvectors $\mathbf{v}$, and $\mathbf{D}$ holds the eigenvalues λ in its diagonal. Since the most important eigenvectors have large corresponding eigenvalues, $\mathbf{D}$ is sorted in order of decreasing λ, and then $\mathbf{v}$ (the columns of $\mathbf{V}$) are sorted accordingly. This way, it is possible to select the m number of $\mathbf{v}$'s with the largest corresponding λ (or select a threshold for λ and choose the $\mathbf{v}$ with λ above this threshold). The remaining $\mathbf{v}$'s represent the transformation matrix $\mathbf{P}$ that is desired.

To perform the dimensionality reduction, the data matrix $\mathbf{X}$ is projected onto the new basis $\mathbf{P}$, which consist of the n extracted eigenvectors. The resulting matrix $\mathbf{Y} = \mathbf{P}^\top\mathbf{X}$ will be of size $M \times n$ instead of $M \times N$. The calculated $\mathbf{P}$ will define a subspace that minimizes the reconstruction error $||\mathbf{X} - \mathbf{P}\mathbf{P}^\top\mathbf{X}||$. Ergo, PCA does not take any class labels into account, it simply treats data points equally, maximizing the variance.

2.5.2 PLS

PLS is very powerful, since it can handle high-dimensional data and take the class labels into account. It is constructed to model the linear relation between two datasets by means of *score vectors*, which are also referred to as latent vectors or components, as they represent variables that are not measured directly. The datasets are usually a set of predictor and response variables, or features and class labels. The score vectors are *projections* of the datasets, and are created by maximizing the covariance between these datasets. There are a few variants of PLS available. The SIMPLS method [13] is the one used in this thesis, and is therefore the one described here. [45]

With one block of m number of N-dimensional predictor variables, stored in a zero-mean $m \times N$ matrix $\mathbf{X}$, and one block of m number of M-dimensional response variables, stored in a zero-mean $m \times M$ matrix $\mathbf{Y}$, PLS models $\mathbf{X}$ and $\mathbf{Y}$ as follows [45]

$$\begin{cases} \mathbf{X} = \mathbf{T}\mathbf{P}^\top + \mathbf{E} \\ \mathbf{Y} = \mathbf{U}\mathbf{Q}^\top + \mathbf{F} \end{cases}, \tag{2.9}$$

where $\mathbf{T}$ and $\mathbf{U}$ are $m \times p$ matrices containing the p extracted score vectors for $\mathbf{X}$ and $\mathbf{Y}$. $\mathbf{P}$ is a $N \times p$ matrix and $\mathbf{Q}$ a $M \times p$ matrix, and they contain the loadings, which describe the linear relation of PLS components that models the original predictor- and response variables respectively. $\mathbf{E}$ and $\mathbf{F}$ are matrices ($m \times N$ and $m \times M$ respectively) containing the residual errors. [45]

The optimal score vectors $\mathbf{t}$ and $\mathbf{u}$ can be found through the construction of a set of weight vectors $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_p]$ and $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_p]$ such that [45]

$$[cov(\mathbf{t}, \mathbf{u})]^2 = [cov(\mathbf{X}\mathbf{w}_i, \mathbf{Y}\mathbf{c}_i)]^2 , \tag{2.10}$$

where $cov(\mathbf{t}, \mathbf{u})$ is the sample covariance between $\mathbf{t}$ and $\mathbf{u}$. In the case that Y is of size $n \times 1$, the formulation can be rewritten as [45]

$$[cov(\mathbf{t}, \mathbf{u})]^2 = max_{|\mathbf{w}|=1}[cov(\mathbf{X}\mathbf{w}_i, \mathbf{y})]^2 . \tag{2.11}$$

The solution to the problem described by (2.11) can be found though singular value de-

composition of the covariance between $\mathbf{X}$ and $\mathbf{Y}$. The weight $\mathbf{w}_i$ will correspond to the first singular vector, and the score vector $\mathbf{t}$ can be calculated as $\mathbf{t} = \mathbf{X}\mathbf{w}_i$. By iterating p times, $\mathbf{W}$ and $\mathbf{T}$ are obtained. The obtained weights $\mathbf{W}$ can then be used as a projection basis, since the score matrix $\mathbf{T} = \mathbf{X}\mathbf{W}$. The chosen number p of score vectors will determine the number of dimensions for the projection ($\mathbf{T}$) of $\mathbf{X}$. [27]

3

Deep Motion Features for Tracking

The SRDCF tracking framework is built to handle hand-crafted features, and only one type of feature can be extracted when running the tracker. In order to investigate the impact of adding deep motion features, several adjustments on the framework were needed. The tracker was rebuilt to both be able to fuse two or more feature types, and to handle optical flow images and extraction of deep motion features. This process is described in sections 3.1 and 3.2. The implementation of the dimensionality reduction techniques is described in section 3.3.

3.1 Convolutional Neural Networks

This section provides a description of which networks and layers have been used for extraction of the deep appearance and motion features. The section is divided according to these two feature types. The choice was made to use two pre-trained networks, and not to train a single network that could handle both types of features, see section 1.4. In all cases when selecting a convolutional layer, the activations after the following ReLU-operation were used as feature maps.

3.1.1 Deep Appearance Features

The network used to extract appearance features is the imagenet-vgg-verydeep-16 network [52], with the MatConvNet library [55]. This network contains 13 convolutional layers, and evaluations were performed using features from both a shallow and deep layer. For the shallow appearance layer, the activations from the fourth convolutional layer were used. This layer consists of a 128-dimensional feature map, and has a spatial stride of 2 pixels compared to the input image region. The deep layer is chosen as the activations after the ReLU-operation after the deepest convolutional layer. This layer consists of 512 feature channels with a spatial stride of 16 pixels. The first row in figure 2.5 shows exam-

ples of features from the shallow layer (first sub-row) and deep layer (second sub-row) of the appearance network.

3.1.2 Deep Motion Features

The motion features are extracted using the method described by [5], who used the approach for action recognition, as mentioned in section 2.4.3.

First the optical flow is calculated for each consecutive pair of frames, according to the approach in section 2.4.3, through use of the algorithm provided by [4]. The motion in the x- and y-directions form a 3-dimensional image together with the magnitude of the flow. The values are then adjusted to the interval $[0, 255]$ to fit the pixel intensities of the image. These calculations were made offline, i.e. not during tracking. The reason for this is that the method chosen to calculate the optical flow takes an average of 6.92 seconds per frame[1].

To extract deep motion features from the optical flow images, the pre-trained motion network by [21] is employed. This network have been trained for action recognition purposes using the UCF101 dataset [53], and contains five convolutional layers. Here, different layers were evaluated before selecting the layer most complementary to the appearance features. The result of these evaluations, which can be seen in section 4.2, shows that the deepest convolutional layer provide the most successful results. This layer gives a resulting feature map with 384 dimensions and spatial stride of 16 pixels.

An example of an optical flow image is displayed in the second row of figure 2.5, together with some of the feature-channels from a shallow and the deep (first and second sub-row respectively) layer of the network.

3.2 Fusing Deep and Hand-crafted Features

The implemented framework is based on learning an independent SRDCF model, according to section 2.2, for each extracted feature map. That is, one filter f_j is learned for each type of feature j.

3.2.1 Feature Extraction

In each new frame k, new training samples $x_{j,k}$ are extracted for every feature type j at the same image region, which is centered at the estimated target location. The training region is quadratic with an area equal to 5^2 times the area of the target box. To extract the deep motion features, the pre-calculated optical flow image corresponding to the current frame is used. In the evaluations, different combinations of feature types are evaluated, see section 4.2. As an example, if three different feature maps are used, e.g. HOG, deep appearance and motion, the training samples for frame k would be: HOG $x_{1,k}$, deep appearance $x_{2,k}$ and the deep motion $x_{3,k}$.

Because the feature maps have different dimensionality size d_j and spatial resolutions, they will also have different spatial sample size $M_j \times N_j$ for each feature type j. Each fea-

[1] Based on calculating the optical flow on 50 frames and calculate the average time per frame.

ture type j is assigned a label function $y_{j,k}$. As in section 2.2, $y_{j,k}$ is a sampled Gaussian function of size $M_j \times N_j$, with maximum centered at the estimated target location.

3.2.2 Training

During training of the correlation filter, the SRDCF objective function (2.2) is minimized independently for each feature type j. This is performed by first transforming (2.2) to the Fourier domain using Parseval's formula, and then applying an iterative sparse solver, similar to [10] as described in section 2.2.

3.2.3 Detection

To detect the target in a new frame, a feature map z_j is extracted for each feature type j using the procedure described in section 3.2.1. The feature map is centered at the estimated location of the target in the previous frame. The filters f_j, which were learned during the previous frame, can then be individually applied to each feature map z_j. The filters are applied at five different image scales with a relative scale factor of 1.02, similar to [10, 33], in order to estimate the size of the target.

Generally, x_j and z_j are created with a stride greater than one pixel, which will lead to the resulting confidence scores $S_{f_j}(z_j)$ of the target being computed on a coarser grid. The size of the confidence scores are $M_j \times N_j$, and the spatial resolution differ for each feature type j. To be able to fuse the confidence scores from each filter f_j, and obtain a resulting pixel location, the scores from each filter need to be interpolated to a pixel-dense grid. Then, the scores can be fused by computing the average confidence value at each pixel location.

The interpolation is performed in the Fourier domain by using trigonometric polynomials and utilizing the complex exponential basis functions of the DFT. Because the filters were optimized in the Fourier domain, the DFT coefficients of each filter $\hat{f}_j$ are already at hand. The DFT coefficients of the confidences can be obtained using the DFT convolution property as follows

$$\widehat{S_{f_j}(z_j)} = \sum_{l=1}^{d_j} \hat{z}_j^l \cdot \hat{f}_j^l. \tag{3.1}$$

Then, the Fourier interpolation can be implemented by zero-padding the DFT confidence coefficients to desired resolution and then apply inverse DFT. Let $\mathcal{P}_{R\times S}$ be the padding operator, which pads the coefficients $\widehat{S_{f_j}(z_j)}$ to the size $R \times S$ by adding zeros at the high frequencies. The desired size is obtained by setting $R \times S$ to the size (in pixels) of the image patch that is used during extraction of the feature maps. Furthermore, let the inverse DFT operator be denoted $\mathscr{F}^{-1}$, and the number of feature maps to be fused J. Then, the fused confidence scores s are can be computed in the following manner

$$s = \frac{1}{J} \sum_{j=1}^{J} \mathscr{F}^{-1} \left\{ \mathcal{P}_{R\times S} \left\{ \widehat{S_{f_j}(z_j)} \right\} \right\}. \tag{3.2}$$

By replacing $\widehat{S_{f_j}(z_j)}$ according to (3.1), and using the linearity of $\mathscr{F}^{-1}$, the previous equation can be rewritten as:

$$s = \frac{1}{J}\mathscr{F}^{-1}\left\{\sum_{j=1}^{J}\mathcal{P}_{R\times S}\left\{\sum_{l=1}^{d_j}\hat{z}_j^l \cdot \hat{f}_j^l\right\}\right\}. \tag{3.3}$$

The final target location is found at the maximum of s.

3.3 Dimensionality Reduction

During initialization of the tracker, i.e. while running the first frame, the original dimensionality size d_j of each feature map x_j is compared to the desired size d_j' for each corresponding feature type j. If $d_j > d_j'$, dimensionality reduction will be performed. Using the first frame, a projection matrix $\mathbf{P}$ (PCA) or $\mathbf{W}$ (PLS) is calculated for each feature type j. The same matrix is then used on the extracted feature maps $x_{j,k}$ (training) and $z_{j,k}$ (detection) for each frame k in the remaining frames of the sequence.

To perform the dimensionality reduction, each feature map is first reshaped from the original structure $M_j \times N_j \times d_j$ to a $M_j N_j \times d_j$ matrix $\mathbf{X'}$, with rows corresponding to pixels and columns to feature types. In other words, each row corresponds to the features of a particular pixel in the feature map. These rows are the predictor variables, which are centered by column-wise extracting the mean, resulting in the zero-mean matrix $\mathbf{X}$.

For dimensionality reduction using PCA, the orthonormal projection matrix $\mathbf{P}$ was created according to section 2.5.1. When using PLS, the projection matrix $\mathbf{W}$ was created with Matlab's build in function `plsregress`, which works in accordance to section 2.5.2. However, the initially received $\mathbf{W}$ does not represent an orthonormal basis. This is handled by calculating and the norm of $\mathbf{W}$, and then achieve an orthonormal basis by (element-wise) division with the norm.

After dimensionality reduction is performed, the projected data is reshaped back to the structure $M_j \times N_j \times d_j'$ and represent the projection of z_j or x_j.

4

Evaluations

This chapter covers the most important evaluations performed during this thesis work. The chapter is divided into four parts, three of which contain different categories of evaluations. The first section describes the evaluation methodology, the metrics used for evaluation scoring, and some information about the datasets. Section 4.2 presents the tests made to investigate the impact of employing deep motion features. Following this are the dimensionality reduction evaluations in section 4.3, where the two methods PLS and PCA are compared. In each of these tests, the best performing tracking setup is taken to the next stage in the evaluations. Finally, in section 4.4, comprehensive tests are performed with comparisons to current state-of-the-art tracking methods.

4.1 About the Evaluations

This section provides some information about how the metrics for the evaluations are defined and what datasets are used. First, a description of how the evaluations are performed is presented.

4.1.1 Evaluation Metrics

The evaluation results are compared using two different metrics. The first is the *overlap precision* (OP), which is defined as the percentage of frames in a video where the intersection-over-union overlap between the ground truth and the estimated center location of the target exceeds a certain threshold b. For a video with N number of frames, the OP is computed as follows:

$$\mathrm{OP}(b) = \frac{1}{N}\left|\left\{t : \frac{|\hat{B}_t \cap B_t|}{|\hat{B}_t \cup B_t|} \geq b\right\}\right|, \quad 0 \leq b \leq 1, \tag{4.1}$$

where t is the frame number, B_t denotes the ground truth box location of the target and $\hat{B}_t$ the estimated box location. The OP at the threshold $b = 0.5$ corresponds to the PASCAL[1] criterion. One OP (at threshold 0.5) is obtained for each video in the dataset used, and the *mean OP* over all videos from each dataset is calculated and used as ranking scores in order to compare different trackers.

In the graphs presented in the evaluation sections, the mean OP is plotted over the range of thresholds $b \in [0, 1]$, which is referred to as a *success plot*. The second evaluation metric is the *area-under-the-curve* (AUC), which is computed from the success plot. The AUC provides an indication of the robustness of the tracker. The ranking score (AUC) is displayed in brackets after the name of each tracking method. Further details about the evaluation metrics can be found at [57].

The VOT2015 dataset evaluation process re-initializes the tracker after each failure, as mentioned in section 4.1.3. The amount of failures for each videos is used to describe the *robustness* of the tracker. The VOT2015 results are reported in terms of the robustness and accuracy (OP) individually for each video, and as an average over all videos in the dataset.

For all datasets, only the average over all videos in the current dataset will be presented in the results.

4.1.2 Datasets

Comprehensive experiments were performed on three challenging benchmark datasets: OTB-2015 [57] with 100 videos, Temple-Color [34] with 128 videos, and VOT2015 [30] with 60 videos. These datasets are all annotated with ground truth of the target object, so that the overlap-precision (OP) can be calculated in accordance to section 4.1.1.

The OTB-2015 dataset consists of a mix of 100 color and gray-scale videos, and Temple-Color 128 color videos. VOT2015 contains 60 challenging color sequences compiled from a set of more than 300 videos as part of the VOT2015 challenge, where state of the art methods are put to the test in a competition. The performance in the challenge is measured both in terms of accuracy and robustness (failure rate), as described in section 4.1.1. For more information and challenge rules, see `http://votchallenge.net`.

The datasets challenge the trackers by containing several different situations. The OTB-2015 dataset include annotation of 11 challenging attributes: illumination and scale variation, occlusion, deformation, motion blur, fast motion, in-plane and out-of-plane rotation, out-of-view, background clutter, and low resolution. In order to identify strengths and weaknesses of the proposed tracker, an attribute-based analysis is performed on the OTB-2015 dataset using the annotations. The results will be displayed in success plots corresponding to the different attributes.

[1] PASCAL stands for Pattern Analysis, Statistical modeling and Computational Learning, and is a Network of Excellence funded by the European Union: `http://www.pascal-network.org/`

4.1.3 General Evaluation Methodology

The evaluations include running the proposed tracker on one or more of the described datasets: OTB-2015 [57], Temple-Color [34], and VOT2015 [30].

During the first frame, the tracker is initialized and given the current position and size of the target box. For the OTB-2015 and Temple-Color datasets, the tracker estimates the location for each of the subsequent frames, which are marked with a ground truth target box used to calculate the different evaluation metrics. The same procedure is repeated for all sequences in the dataset, and results for all videos are available as an average for the dataset or individually for each video. The evaluation for VOT2015 is slightly different. Whenever the target is lost, the tracker is noted with a failure and is re-initialized a few frames after the failure.

4.2 Deep Motion Features for Tracking

The first aim of this theses is to investigate the use of deep motion features in tracking. First, the SRDCF framework (see section 2.2) was adjusted to be able to handle multiple feature representations, according to section 3.2, and the CNNs and layers used for feature extraction were chosen as in section 3.1.

The first evaluations include investigating which layers from the motion network are most useful and complementary to appearance features. These evaluations are presented in section 4.2.1. Then, a few experiments were performed to see if the optical flow images could be improved to get a better result, which is evaluated in section 4.2.2.

When the initial experiments were completed, the so far most successful settings were used in the investigation of the impact of adding deep motion features to a tracker employing appearance features. These evaluations are presented in section 4.2.3. All of these evaluations were performed using the OTB-2015 dataset.

4.2.1 Selecting Motion Layers

To investigate which, and how many, layers from the motion network would be best suited for the task, a few initial experiments were performed. HOG features were used to represent appearance information, due to their success in recent tracking frameworks [10, 8, 25].

Table 4.1 presents an overview of the evaluation settings. Features were extracted (after the ReLU-operation) from the first, fourth and fifth (deepest) convolutional layer in accordance to section 3. Several more motion layer combinations were tested, but since the results were similar and not particularly interesting, they are not shown here. The table explains how the layers were used/combined in the six experiments, which are referred to as A1-A6. HOG features were, as mentioned, used for A1-A6. A1 represent the baseline result, i.e. without any motion features, which is used in comparison the other results.

Table 4.1: *Overview of the evaluation of extracting deep motion features from different layers of the deep CNN that handles optical flow images. HOG features were employed in all experiments A1-A6, and the first column in the table shows the different motion layers used. Here, the layer number refers to the convolutional layer in the network used to extract features. The x marks which feature types are included in the different experiments A1-A6. In A1, only HOG features were used in order to get a baseline comparison for the evaluation, which is why the column is empty in the table.*

Feature type	A1	A2	A3	A4	A5	A6
Motion layer 1		x				x
Motion layer 4			x		x	
Motion layer 5				x	x	x

Results

The result of the deep motion layer evaluation is displayed in the success plot in figure 4.1, and is summarized in terms of mean OP and AUC in table 4.2.

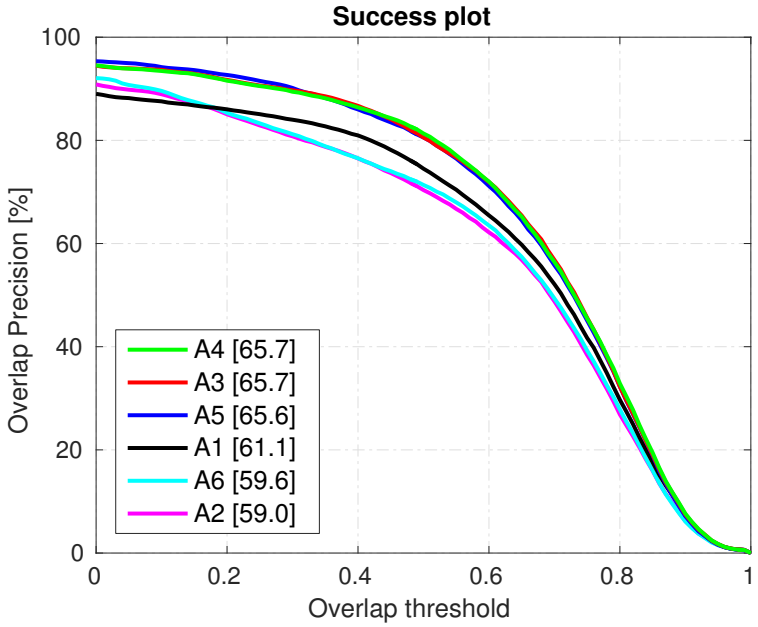

Figure 4.1: *Success plot with the results from the evaluation using different deep motion layers. The baseline A1 is using only HOG features, while the remaining A2-A6 are combinations of HOG and deep motion features extracted from different layers in the deep motion network. A1-A6 are defined in table 4.1. The legend shows the trackers in order of highest AUC, which is the number displayed in brackets after the experiment name.*

Table 4.2: *Evaluation results of trackers using HOG features as baseline (A1), and adding deep motion features from the first, fourth and fifth convolutional layers according to table 4.1, where the experiment settings A1-A6 are defined. The results are presented in form of mean OP and AUC in percent, and the two best results are shown in red and blue font respectively.*

	A1	A2	A3	A4	A5	A6
Mean OP	74.5	70.4	80.6	**81.3**	*80.6*	71.4
Mean AUC	61.1	59	*65.7*	**65.7**	65.6	59.6

Discussion

The success plot in figure 4.1 shows that the experiment settings A3, A4 and A5 (defined in table 4.1) are superior to the others. These settings include HOG features combined with motion features from the two deepest layers in the motion CNN. The shallower layers does not seem to provide any important information to the tracker, as they result in a score that is lower than the baseline, employing only HOG.

Table 4.2 shows that the best results are given when only using setting A4 (defined in section 4.1), i.e. HOG and features from the deepest motion layer. The mean AUC for this setting is 65.7 %, which is an improvement of +4.6 percentage points compared to using only HOG (A1), and the mean OP increased to 81.3 %, +6.8 percentage points compared to A1. The top three trackers have very similar result, so using any of the two deepest layers would work for future evaluations. Here, the deepest motion layer will be employed in future evaluations, since it achieved the highest score.

4.2.2 Using Mean and Median Subtraction

An attempt was made to enhance the optical flow images and see if it would affect the tracking results in a positive manner. The idea was to make the target stand out more in cases where the camera is moving, as this will cause the background to have increased magnitude in the optical flow image. Examples of this can be seen in figure 4.6, and it is discussed further in the discussion section of this evaluation (section 4.2.3). By calculating and then subtracting the mean or median of the image, the background magnitude would be reduced, which could provide better images to extract motion features from.

In the experiment, mean and median were calculated in three different ways. In the first case, they were calculated over the entire image. In the second case, they were calculated in the image region outside the estimated target box, and in the last case outside a box centered at the same position as the target box, but with twice the size. For simplicity, the cases are named A, B, and C respectively, and are visualized in figure 4.2. The calculations were performed during tracking, before the motion features were extracted.

The tracker with the so far best results were used for the experiments, which is utilizing HOG together with deep motion features from the deepest layer in the motion network. The evaluation was performed on the OTB dataset.

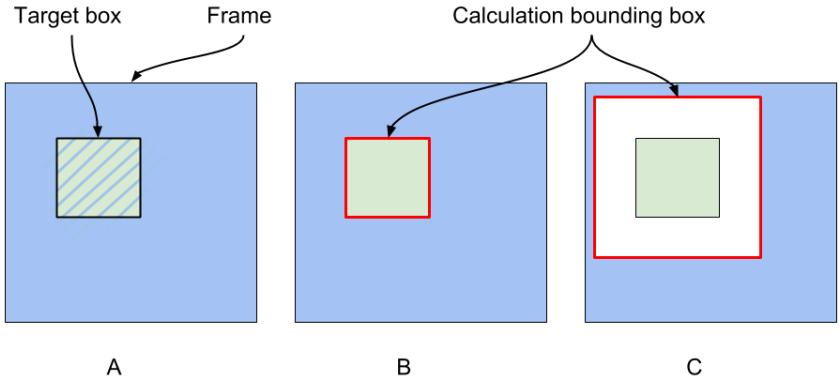

Figure 4.2: *Visualization of the three methods employed in mean and median calculation. The blue area represents the area used for the calculations in the three cases. In case A, the area inside the green box, which represents the target box, is also included. In cases B and C, the blue area is kept outside the red bounding box.*

Results

The resulting mean OP and AUC from the evaluations using different settings of mean or median subtraction are presented in figure 4.3 and is summarized in terms of mean OP and AUC in table 4.3.

Discussion

The results in table 4.3 states that it is not beneficial to use any of the suggested kinds of mean or median subtraction in the tracking process. The success plot in figure 4.3 shows that the results are fairly similar, but that the baseline achieves top score. Extracting the mean according to method B (displayed in figure 4.2), i.e. outside the target box, provided the best result of the different methods evaluated. It achieved 83.3 % mean OP and 66.9 % mean AUC. This is -0.6 and -0.5 percentage points lower respectively compared to the baseline, which received 84.1 % mean OP and 67.4 % mean AUC.

The results were a bit surprising, since the optical flow images were meant to be enhanced by the procedure, which removes the background movements. Contrary, it seems that using such an approach destroys information rather than assisting in extracting it. It is not entirely unreasonable however. If the object is, for example, moving in the same direction as the background in relation to the camera, the object would have similar optical flow values as the background. Subtracting the mean or median would then leave a weaker output for the object compared to before, which could affect the feature extraction negatively. Also, it cannot be assumed that the target box always is located on the correct position, i.e. where the target actually is. Performing the proposed approach in such situations would likely not give pleasant results.

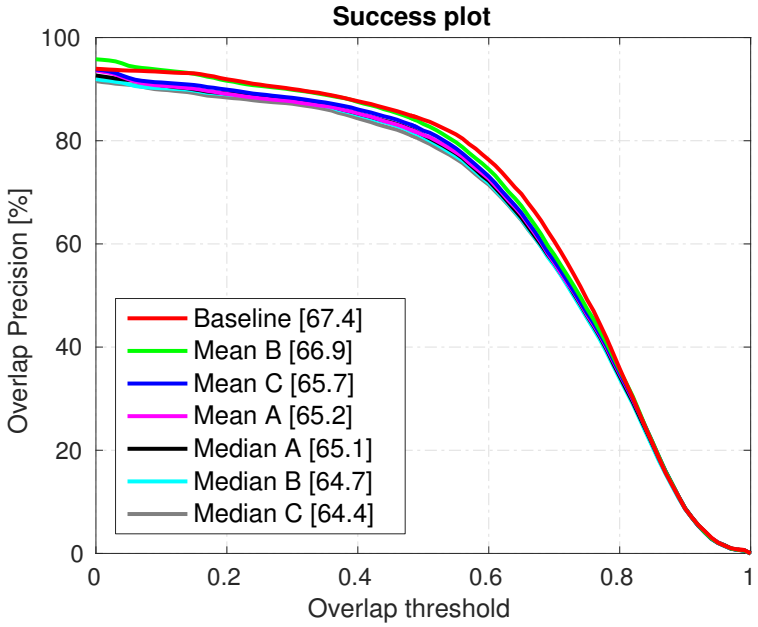

Figure 4.3: *Success plot with the results from the evaluation using mean and median subtraction on the optical flow images prior to feature extraction. A, B and C specify the method of calculating the mean and median, which is explained in section 4.2.2 and visualized in figure 4.2. The legend shows results in order of AUC score, which is the number shown in brackets after the experiment name.*

Table 4.3: *Resulting mean OP and AUC from the evaluation using mean and median subtraction on optical flow image. A, B and C denotes how the mean or median was calculated in the optical flow images, which is explained in section 4.2.2 and visualized in figure 4.2. The baseline represent the results without using mean or median subtraction. The two best results are displayed in red and blue font respectively.*

	Baseline	Mean A	Mean B	Mean C	Median A	Median B	Median C
Mean OP	**84.1**	81.1	*83.3*	82	81.1	80.7	80
Mean AUC	**67.4**	65.2	*66.9*	65.7	65.1	64.7	64.4

4.2.3 Impact of Deep Motion Features

To investigate the impact of adding deep motion features, several experiments are performed using different parameter settings.

Seven feature settings, referred to as B1-B7, were used both together with, and without deep motion features. An overview of the experiments B1-B7 is shown in table 4.4. Using this combination of experiment settings will make it easy to compare if and when the results of the tracker improve. The deep and shallow layer features were extracted according to section 3.1.

Table 4.4: *Overview of the evaluation regarding the impact of deep motion features. The x marks which feature types are included in the different experiments B1-B7. All experiments were executed two times: with and without deep motion features.*

Feature type	B1	B2	B3	B4	B5	B6	B7
HOG	x				x	x	x
Shallow appearance (RGB(s))		x		x	x		x
Deep appearance (RGB(d))			x	x		x	x

Results

Figure 4.4 shows the success plot for the evaluation. The resulting mean OP and AUC from the 14 different feature settings are displayed in table 4.5 below. In the table, shallow and deep appearance features are denoted RGB(s) and RGB(d) respectively.

Discussion

The results table shows that adding deep motion features consistently improves the performance of the tracker for all the evaluated combinations B1-B7 (defined in table 4.4). The mean OP and AUC over all videos increase significantly when adding motion fea-

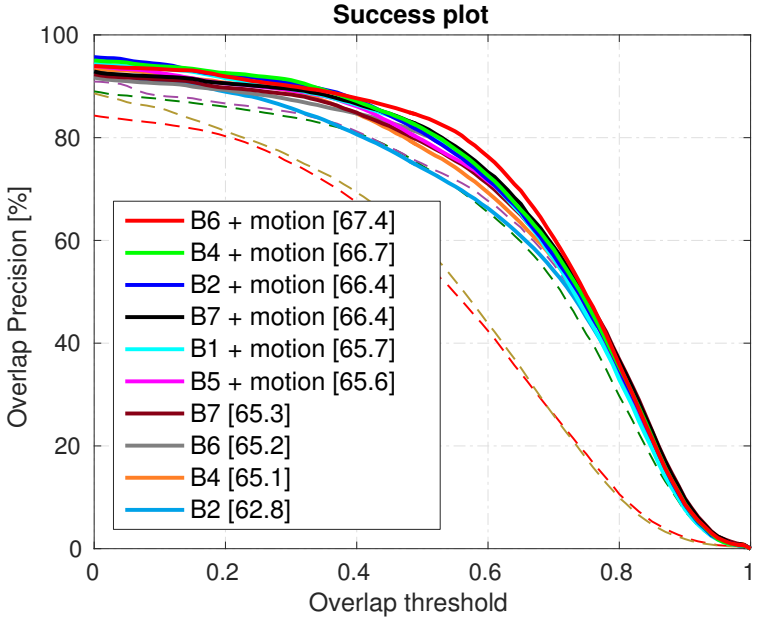

Figure 4.4: *Success plot with the results from the impact of deep motion features evaluation. B1-B7 are the experiment settings, which are explained in table 4.4. B1-B7 were used with and without deep motion features, notated as motion in the legend, which shows the ten best results in order of AUC score. The dashed lines represent the four least successful results.*

Table 4.5: *Results from evaluation using different combinations of hand-crafted (HOG), deep appearance (RGB) and deep motion features. B1-B7 represent the evaluation settings, as displayed in table 4.4 Shallow and deep layers of the appearance network are denoted RGB(s) and RGB(d). Results are reported in terms of mean OP and AUC, and the two best results are shown in red and blue font respectively.*

		B1 HOG	**B2** RGB(s)	**B3** RGB(d)	**B4** RGB(s+d)	**B5** HOG+RGB(s)	**B6** HOG+RGB(d)	**B7** HOG+RGB(s+d)
Mean OP	Without motion features	74.5	74.1	56.3	78	74.9	80.7	79.1
	With motion features	81.3	81	58.9	81.1	79.5	**84.1**	*82.2*
Mean AUC	Without motion features	61.1	62.8	48.5	65.1	62.6	65.2	65.3
	With motion features	65.7	66.4	49.7	*66.7*	65.6	**67.4**	66.4

tures. The best result was given for B6 with deep motion features, which includes HOG and deep appearance (RGB(d)) features. This setting resulted in 84.1 % mean OP, an improvement of +3.4 percentage points compared to B6 without deep motion features, and +1.9 percentage points higher than the next best result (B7 with motion features), which also included the RGB(s) layer. The mean AUC for B6 was 67.4 %, an improvement of +2.2 points compared the result of B6 when not using deep motion features, and +0.7 compared to the next best setting (B4 with motion features), which employs RGB(s) and RBG(d) features.

The biggest improvement was seen in B1, using HOG, where the mean OP increased by +5.7, and in B2, using RGB(s), where the mean AUC increased by +4.6. This leads to another interesting observation. Using HOG (B1), or a shallow appearance layer (RGB(s), B2), alone result in approximately the same mean OP and AUC as using a combination of HOG and RGB(s) (B5). This implies that these two feature representations are not particularly complementary to each other. However, when adding deep motion features to either of these feature representations the result improves significantly. A clear indication that deep motion features indeed does contain complementary information to both deep and hand-crafted appearance features.

A few videos where adding motion features appeared to give particularly good results were *Box*, *Coupon* and *Skiing*. A few frames from these sequences are shown in figure 4.5. These videos contain fast motion, occlusion and rotation of the target (first row), distractors with similar appearance (second row) and distant objects (third row). Another difficult scenario where motion features provided improved result is motion blur. The sequence *Human7* includes substantial amounts of camera movement. This is a very challenging case for the tracker, since a shaking camera will cause motion blur, and fast changes in target location. A few frames from the video can be seen in figure 4.6 (first row), together with the corresponding optical flow images (second row).

As stated in section 2.4.3, the optical flow captures the motion energy from each consecutive pair of frames. If the camera is moving with a certain velocity v_{xy}, it will be equivalent to the world moving with $-v_{xy}$, corresponding to a certain color in the optical flow image. As the camera changes direction, so will the color of the background in the optical flow image, as seen in the image.

If an object is moving with a different velocity compared to the background, it will show up in another color. Whereas the original frames contain excessive amounts of unneces-

sary information about the background, the optical flow captures important information about the moving target object.

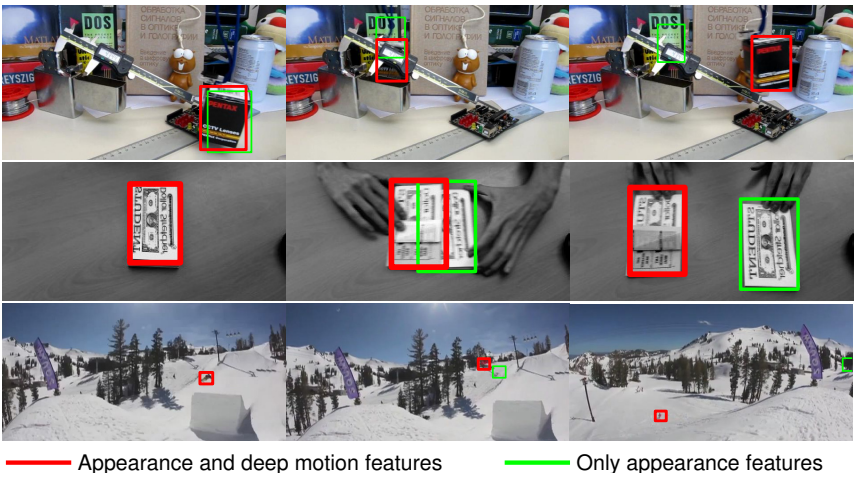

Figure 4.5: *A comparison of using a combination of hand-crafted (HOG) and deep appearance features (green), and using a fusion of appearance and deep motion features (red). A few frames with tracking results are shown for three example sequences: Box, Coupon and Skiing. The red box is on the correct target in each frame.*

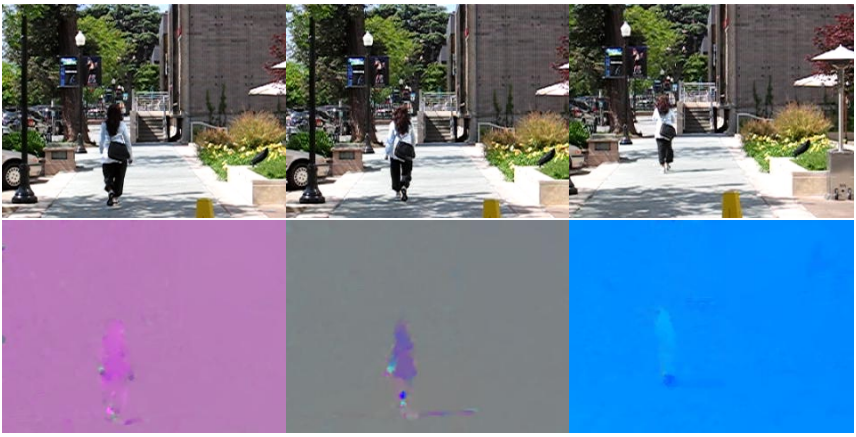

Figure 4.6: *Frames from sequence Human7 and corresponding optical flow images. The optical flow images have different background color due to very shaky camera movements. The camera moving to the right will result in the entire background moving to the left in the video, resulting in a homogeneous optical flow. As the woman is walking, she will have a different direction than the background in the video, and thereby display a different optical flow.*

The example back in figure 2.5, showing an image, its optical flow equivalent, and examples of extracted appearance- and motion features, gives good understanding of how the feature representations differ when it comes to deep appearance and motion features. When comparing the activations from the shallow layers of both networks, it is clear that the appearance features have a lot of distractions to handle, whereas the deep motion features are concentrated where the object actually is. Features from deeper layers provide more high-level information, and trying to interpret them can therefor be difficult, but given the results, and using the information stated about how the images and features differ, it can be concluded that the deep motion features clearly contain information complementary to hand-crafted as well as deep appearance features.

4.3 Dimensionality Reduction

To evaluate and compare the performance of the two dimensionality reduction techniques, several experiments were performed using different settings and evaluating the result on the OTB-2015 dataset. The baseline is the tracker with the so far best results, which employs HOG together with deep appearance and motion features.

Four initial experiments were performed, using both PCA and PLS, to see how the results were affected when reducing the feature dimension for either the deep appearance or motion features. After the results of these experiments had been observed, two additional tests were made, using dimensionality reduction on both deep feature types. An overview of the settings is compiled in table 4.6, in which C1-C4 represent the initial four experiments, and C5 and C6 the following two.

When the evaluation is run on the computer using multiple threads, the tracker is run simultaneously on the different threads. Each run will therefore only be given a single thread, i.e. lower computational power compared to one tracker being run alone using the combined computational power of all threads. Even if the tracker's performance is unaffected in terms of OP, it will receive an unfairly low frame count. Running the tracker with multi threading turned off will provide a more powerful computation, resulting on a more true frame count. Because of this, the baseline and setup providing highest number of frames-per-second (FPS) in the evaluation were re-evaluated on 10 sequences with this

Table 4.6: *Summary of the experiments performed regarding dimensionality reduction. The deep appearance layer is denoted RGB(d). Original size refers to the number of feature dimensions before dimensionality reduction. C1-C6 represent the different settings used in each experiment. In for example C1, the dimension of the deep appearance features are reduced from 512 to 100, while the deep motion features are kept at their original dimension size. The same experiment settings were applied using both PCA and PLS.*

Feature layer	Original size	C1	C2	C3	C4	C5	C6
RGB(d)	512	100	50	512	512	50	50
Deep motion	384	384	384	80	30	80	30

setting to get a better understanding of the efficiency gain.

The tracker is re-trained each frame using the extracted training feature maps. To investigate the gain in required time for training calculations, a timer was inserted in the tracker, which was then run on a sequence of 140 frames. This was performed once for the baseline settings, and then with PLS and PCA on the setup that provided fastest frame count in the evaluation.

4.3.1 Result

The result of the dimensionality reduction evaluation is summarized in table 4.7. The result is shown in means of the mean OP and AUC in percent. To evaluate the gain in speed, the number of frames handled per second is also reported in the table. The success plot is shown in figure 4.7. The "true FPS" evaluation results are found in table 4.8, and the results from the training time evaluation are presented in table 4.9.

4.3.2 Discussion

As seen in table 4.7, both PCA and PLS give successful results. PCA achieved best results in setup C1 (defined in table 4.6), where the RGB(d) feature dimension is reduced from 512 to 100, resulting in mean OP of 85.5 %, +1.4 percentage points better than not using any dimensionality reduction, and AUC of 68 %, +0.6 percent points higher. PLS obtained best results for C3, where the deep motion feature dimension is reduced from 384 to 80, and achieved the best results in total, with mean OP 85.9 %, +1.8 percentage points

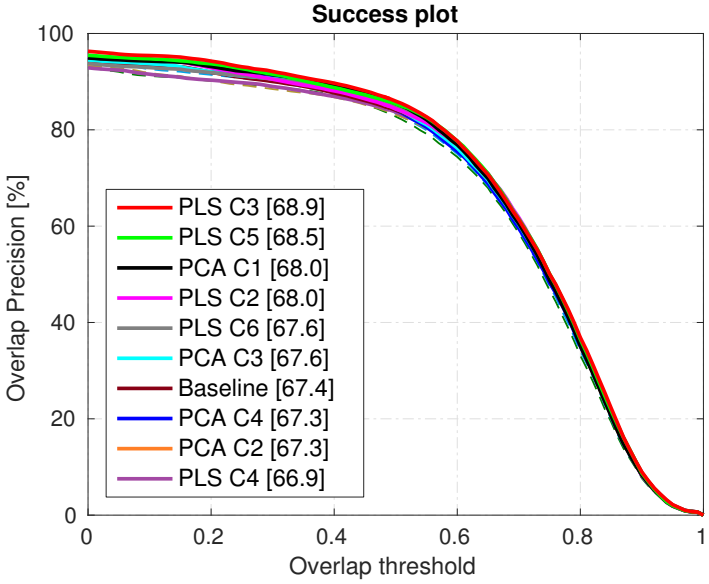

Figure 4.7: *Success plot for the experiments C1-C6 in table 4.6 with PLS and PCA. The legend shows the top ten results in terms of mean area-under-curve (AUC) in percent (shown in brackets after experiment).*

better than not using any dimensionality reduction, and AUC 68.9 %, +1.5 percentage points better. However, the two methods differ very little in terms of performance, which is apparent in the success plot in figure 4.7, where all lines are somewhat close together.

Table 4.7: *The result of the six experiments on PCA and PLS. The baseline (without dimensionality reduction) is also shown for preference. The mean OP and AUC are given in percent, while FPS is the mean number of frames calculated per second. The two best results for each result type is shown in red and blue fonts respectively. The experiments C1-C6 are equal to the ones presented in table 4.6. For clarification, a short description is added under each experiment name. For example, C1 has description "rgb -> 100", which means that the dimension of the deep appearance (RGB(d)) features have been reduced to 100. The deep motion features are denoted "of" (optical flow) in the descriptions.*

		Baseline	**C1** rgb-> 100	**C2** rgb-> 50	**C3** of-> 80	**C4** of-> 30	**C5** rgb-> 50, of-> 80	**C6** rgb-> 50, of-> 30
PCA	**Mean OP**	84.1	**85.5**	84.3	*84.4*	84.1	83.4	82.7
	Mean AUC	67.4	**68**	67.3	*67.6*	67.3	66.6	65.9
	Mean FPS	0.0659	0.0787	0.0783	0.0788	0.0783	*0.0831*	**0.084**
PLS	**Mean OP**	84.1	83.1	84.3	**85.9**	83.7	*85.4*	84.8
	Mean AUC	67.4	66.8	68	**68.9**	66.9	*68.5*	67.6
	Mean FPS	0.0659	0.0778	0.0784	0.0787	0.0781	*0.0833*	**0.084**

Table 4.8: *The true frame count for the baseline tracker and with PLS dimensionality reduction according to the settings in C6 (defined in table 4.6), where the feature dimension is reduced to 50 for RGB(d), and to 30 for deep motion. The total efficiency gain is in comparison to when not using dimensionality reduction. The gain was calculated by taking the difference between the old and new FPS, and dividing with the old.*

	Baseline	**C6 PLS**	**Efficiency gain**
FPS	0.258	0.298	15.50 % faster

Table 4.9: *Training time gained compared to when not using dimensionality reduction. The gain was calculated by taking the difference between the old time required for training, and new training time plus the time required for performing the dimensionality reduction and divide it by the old time.*

	Mean training time	**Calculation time**	**Training time gain**
Baseline	0.2007 s		
PCA	0.0833 s	0.0019 s	57.55 % faster
PLS	0.0836 s	0.0036 s	56.55 % faster

PLS seems to provide better results when applying dimensionality reduction to both feature types, as seen in C5 and C6 , where PLS achieves a mean OP which is respectively +2 and +2.1 percentage points better than when using PCA, and the mean AUC is +1.9 and +1.7 percentage points better. Experiment C5 using PLS, where the motion features are reduced from dimension 384 to 80, and appearance features from 512 to 50 actually achieved the second best results of the evaluation, which is why the tracker with these settings from now on is considered the best tracker.

In table 4.7, it may look somewhat undesirable that the FPS at best resulted in 0.084. It partly has to do with the single thread problem mentioned in the evaluation description. Running the baseline, and PLS C6 (defined in table 4.6) trackers with multi threading turned off resulted in frame count as in table 4.8. Here, the proposed approach runs at 0.298 FPS, compared to 0.258 for the baseline. This means that using PLS, with settings as in C6, provides an efficiency gain of 15.5 %.

The results from investigating the time required for re-training the model in each iteration is seen in table 4.9. Without dimensionality reduction of the training features, the tracker takes 200.7 milliseconds to re-train the model. When using PCA, the training time was reduced to 83.3 milliseconds, and with the time required to perform the actual dimensionality reduction taken into account, this corresponds to an improvement of 57.55 %. PLS is not far behind, with 56.55 % efficiency gain. This also reveals that PLS takes an average of 1.7 milliseconds longer, not much but it still means PLS takes almost twice the time compared to PCA. However, the mean training time is only 0.3 milliseconds behind PCA, and PLS has roughly the same mean FPS as PCA in table 4.7, so the method makes up for the lost time quite efficiently.

As it turns out, the reason for the, in general, low FPS and rather small total efficiency gain is not the large feature dimensions per se, even if utilizing dimensionality reduction does increase the frame count a bit. There are other properties of the tracker that affects the FPS more. For one thing, the feature extraction takes quite a lot of time. This is mainly because of two things: the CNNs require heavy calculations when extracting features, and the scale approximation method employed (see section 3.2.2) does in practice mean that the features have to be extracted five times (one for each scale).

There may also be other factors that could be changed in the tracker to achieve a higher FPS, such as parameter tuning. This was however not investigated further, since the aim here was more to evaluate the two dimensionality reduction techniques rather than achieving a really fast tracker, as mentioned in the scope section (1.3).

Overall, it is beneficiary to use dimensionality reduction, both in terms of efficiency and tracking performance. The two methods are fairly equal in performance, but taking the labels into account and maximizing the covariance seems give PLS a small upper hand. So PLS is determinately a good alternative to PCA for tracking applications.

4.4 State-of-the-art Comparison

In these evaluations the resulting tracker is put to the test. This final proposed tracking framework employs HOG together with deep RGB- and motion features, according to settings B6 (found in table 4.4), where the deep features have been reduced in dimension using PLS according to settings C5 (found in table 4.6).

In order to evaluate the performance of the proposed approach, comprehensive comparisons are made with relevant state-of-the-art trackers. The evaluation is performed on both the OTB-2015 and the challenging Temple-Color datasets. The comparisons include 19 state-of-the-art trackers: LSHT [24], TGPR [20], ASLA [28], EDFT [15], TLD [29], Struck [23], CFLB [19], LCT [37], ACT [9], KCF [25], DSST [8], SAMF [33], DAT [43], MEEM [58], DFT [48], HCF [36], SRDCF [10] and SRDCFdecon [12]. The DeepSRDCF [11], which employs the shallow layer of a deep RGB network in the SRDCF, is also included in the comparison.

An evaluation was also performed on the VOT2015 dataset, where the result is compared with the top 10 participants in the challenge (see section 4.1.2). More information about the datasets is found in section 4.1.2.

4.4.1 OTB-2015

The videos in the OTB-2015 dataset include both color and gray-scale, and they represent several challenging scenarios, and they are labeled accordingly, which enables an attribute based comparison to be made. The scenarios are: illumination and scale variation, motion blur, out-of-view, occlusion, deformation, in-plane and out-of-plane rotation, fast motion, background clutter and low resolution.

Result

Figure 4.8 shows the success plot of the evaluation on the OTB-2015 dataset. The attribute based results are displayed as success plots in figures 4.9 and 4.10. Table 4.10 provide a summary of the mean OP and AUC for the top 13 trackers.

Discussion

The success plot in figure 4.8 shows that the proposed tracker clearly outperforms the state-of-the-art trackers. This is also apparent in table 4.10, where the tracker is seen in red as the best result, with mean OP of 85.4 % and AUC 68.5 %. In the table, the result of the regular SRDCF tracker is seen with mean OP 72.9 % and AUC 60.5 %. The second best results were achieved by the DeepSRCF tracker, which received mean OP 77.3 % and AUC 64.3 %.

This reveals that developing the SRDCF tracker to utilize deep appearance instead of hand-crafted features (deepSRDCF) improved the result by +4.4 percentage points for the mean OP, and AUC by +3.8. However, developing the SRDCF tracker to employ a fusion of hand-crafted, deep appearance, and deep motion features (tracker produced in this thesis) improved the mean OP by another +8.1 percentage points. This score is a massive +12.5 points better than the regular SRDCF. The mean AUC improved by +4.2

points compared to the DeepSRDCF, and with +8.0 percentage points in comparison to the regular SRDCF.

Further more, the proposed tracker achieved best results for all of the 11 challenging attributes. The attributes with most superior results compared to the next best existing tracker were: deformation (+8.1 mean AUC percentage points), out-of-plane rotation (+5.5), out of view (+5.2), and occlusion (+5.2). Example frames including such scenarios can be seen in figure 4.11, where the results of the developed tracker are compared frame-by-frame with state-of-the-art methods.

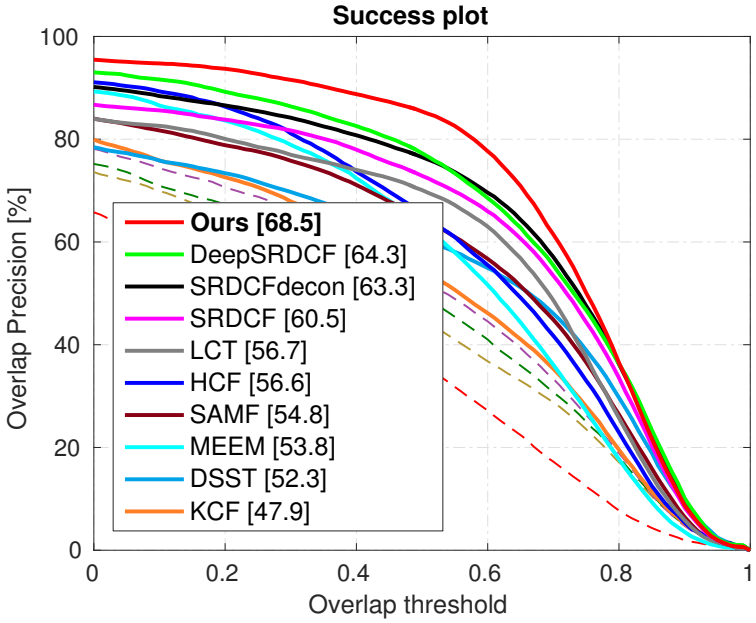

Figure 4.8: *Success plots showing a comparison of the proposed tracker (in red, referred to as Ours) with state-of-the-art trackers on the OTB-2015 dataset. The top 10 trackers are shown in the legend, and the remaining as dashed lines.*

Table 4.10: *Summary of resulting mean OP and AUC for the top 13 trackers in the state-of-the-art evaluation on the OTB-2015 dataset. The proposed tracker of this thesis in referred to as Ours. The two best results are displayed in red and blue fonts respectively.*

	Struck	CFLB	ACT	KCF	DSST	SAMF	DAT	MEEM	LCT	HCF	SRDCF	SRDCFdecon	DeepSRDCF	**Ours**
Mean OP	52.9	44.9	49.6	54.9	60.6	64.7	36.4	63.4	70.1	65.5	72.9	76.5	*77.3*	**85.4**
Mean AUC	46.3	41.4	44	47.9	52.3	54.8	33.6	53.8	56.7	56.6	60.5	63.3	*64.3*	**68.5**

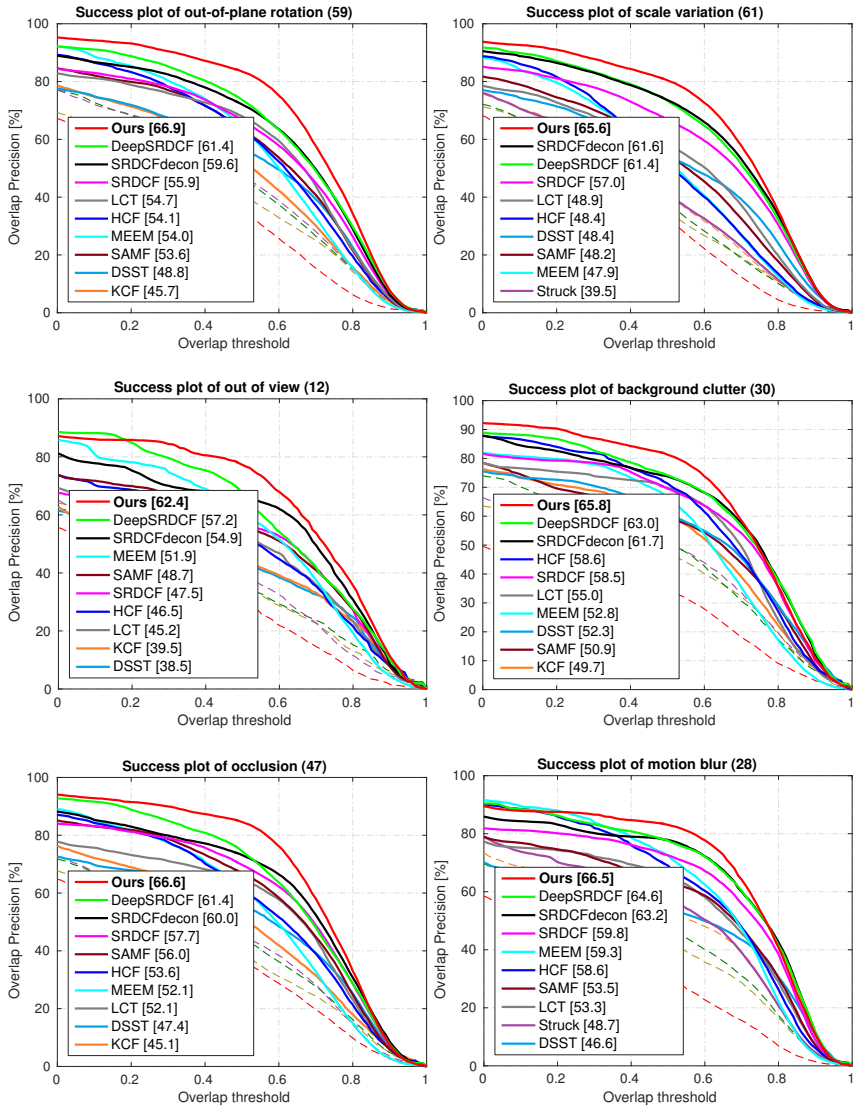

Figure 4.9: *Attribute-based analysis of the proposed tracker (referred to as Ours in the plots) on the OTB-2015 dataset. Success plots are shown for six of the attributes (the remaining five are shown in figure 4.10). For clarity, only the top 10 trackers are shown in each plot, displayed in order of success in the legend. The remaining trackers are shown as dashed lines. The title above each plot indicates the number of videos labeled with the respective attribute in the dataset.*

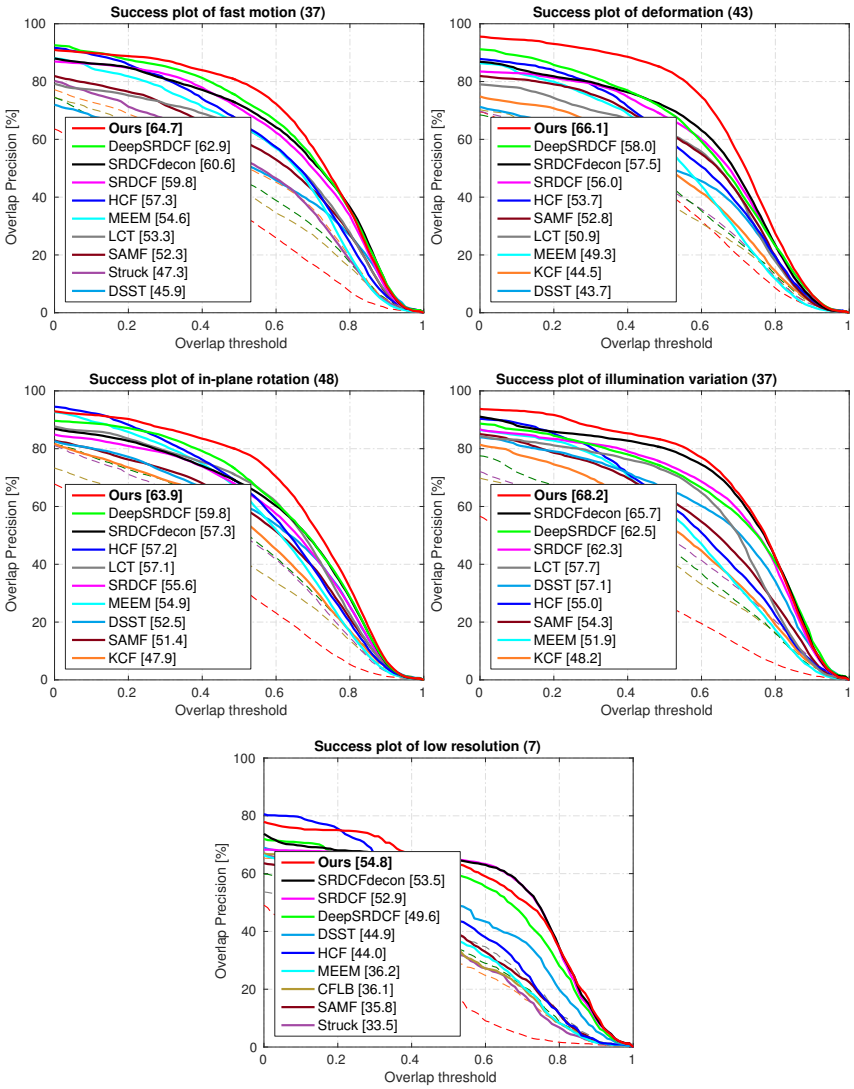

Figure 4.10: *Attribute-based analysis of the proposed tracker (referred to as Ours in the plots) on the OTB-2015 dataset. Success plots are shown for five of the attributes (the remaining six are shown in figure 4.9). For clarity, only the top 10 trackers are shown in each plot, displayed in order of success in the legend. The remaining trackers are shown as dashed lines. The title above each plot indicates the number of videos labeled with the respective attribute in the dataset.*

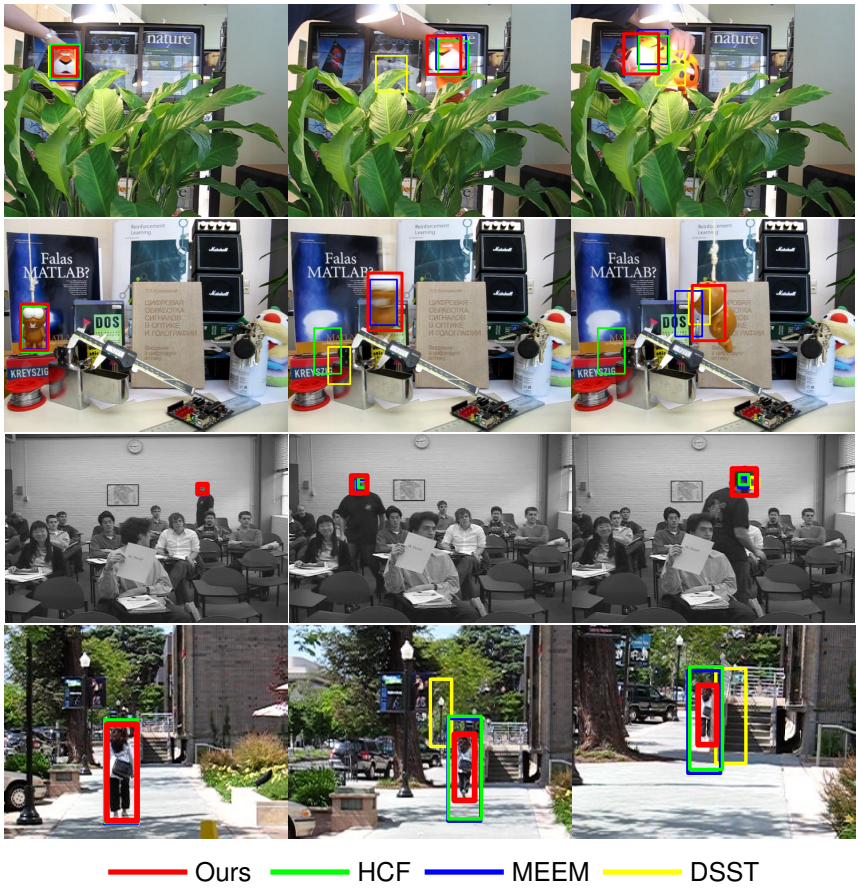

Figure 4.11: *Frame-by-frame comparison with three state-of-the-art trackers on the sequences Tiger2, Lemming, Freeman and Human7. These videos contain scenarios with e.g. occlusions, in- and out-of-plane rotations and blur. The tracker developed in this thesis is referred to as Ours (red).*

4.4.2 Temple-Color

The proposed tracker was evaluated on the Temple-Color dataset, containing 128 labeled color videos with several challenging tracking scenarios.

Result

The result of the evaluation on the Temple-Color dataset is shown in the success plot in figure 4.12, and in terms of mean OP and AUC in table 4.11, together with the result of the 13 state-of-the-art trackers.

Discussion

The success plot in figure 4.12 shows that the proposed tracker is achieving an OP which is better than the other state-of-the-art trackers for almost all threshold values. In table

4.11, it is shown that the proposed approach receives a mean OP of 69.6 %, which is +4.2 percentage points better than the next best tracker and +7.4 compared to the regular SRDCF tracker, which employs only appearance features. The mean AUC achieved was 56.7 %, which is +2.4 compared to the next best tracker, and +5.1 compared to the regular SRDCF.

It is clear that this modified implementation of the SRDCF tracker, utilizing deep motion and appearance features in combination with the hand-crafted appearance features, is very successful on the challenging videos in the Temple-Color dataset. It improves the result compared to the regular SRDCF, and outperforms other state-of-the-art methods.

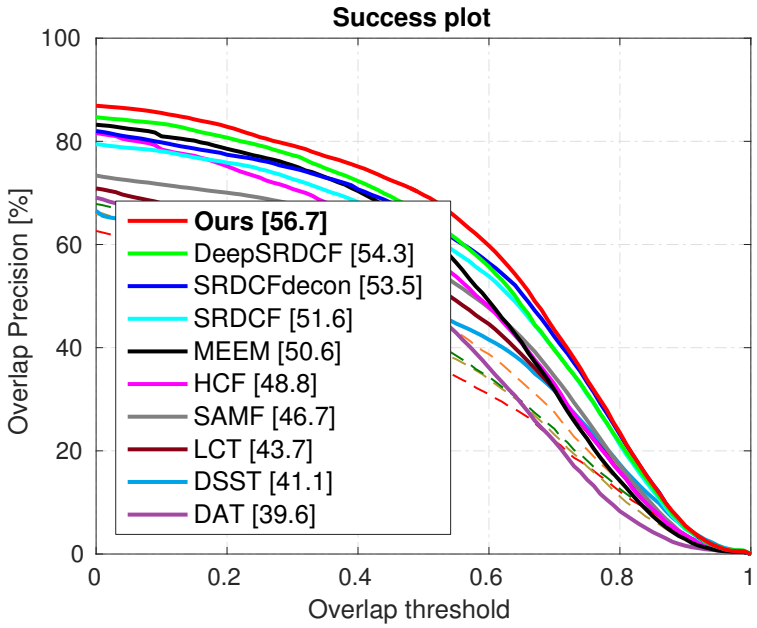

Figure 4.12: *Success plots showing a comparison of the proposed approach (Ours, in red) with state-of-the-art methods on the Temple-Color dataset. The legend shows the top 10 in terms of AUC. The dashed lines represent the trackers that are not in the top ten.*

Table 4.11: *Summary of resulting mean OP and AUC for the top 13 trackers in the state-of-the-art evaluation on the Temple-Color dataset. The proposed tracker of this thesis in referred to as Ours. The two best results are displayed in red and blue fonts respectively.*

	Struck	CFLB	ACT	KCF	DSST	SAMF	DAT	MEEM	LCT	HCF	SRDCF	SRDCFdecon	DeepSRDCF	**Ours**
Mean OP	40.9	37.8	42.1	46.5	47.5	56.1	48.2	62.2	52.8	58.2	62.2	65	*65.4*	**69.6**
Mean AUC	36.6	34.5	37.8	39	41.1	46.7	39.6	50.6	43.7	48.8	51.6	53.5	*54.3*	**56.7**

4.4.3 VOT2015

The evaluation on the VOT2015 dataset was performed at an earlier stage of the thesis, before the dimensionality reduction evaluation, as part of the results for the paper [22]. Due to time limitations, it was not re-evaluated with the new tracker employing the dimensionality reduction.

Result

The result of the evaluation on the VOT2015 dataset is seen in table 4.12, together with the top 10 results from other trackers in the challenge.

Table 4.12: *Comparison with the top 10 state-of-the-art methods on the VOT2015 dataset. The proposed tracker is referred to as Ours. The two best results are displayed in red and blue fonts respectively.*

	S3Tracker	RAJSSC	Struck	NSAMF	SC-EBT	sPST	LDP	SRDCF	EBT	DeepSRDCF	**Ours**
Robustness	1.77	1.63	1.26	1.29	1.86	1.48	1.84	1.24	*1.02*	1.05	**0.92**
Accuracy	0.52	*0.57*	0.47	0.53	0.55	0.55	0.51	0.56	0.47	0.56	**0.58**

Discussion

The results table shows that the proposed tracker achieves a robustness of 0.92 failures per video, and accuracy of 0.58 (i.e. mean OP 58 %). These scores are better than the other top 10 trackers, with 0.1 less failures and +0.1 mean OP (+1 percentage points) compared to the next best trackers.

The proposed tracker outperforms the other methods, though the results are not as superior as in the OTB-2015 or Temple-Color state-of-the-art evaluations. However, since the proposed approach achieved better results using dimensionality reduction, as stated in the evaluation in section 4.3. It is therefore probable that the results on the VOT2015 dataset would improve further using the approach with dimensionality reduction.

5

Conclusions and Future Work

A tracker has been implemented as a modification of the SRDCF framework, enabling the use of a combination of deep and hand-crafted appearance features with deep motion features extracted using optical flow images. The evaluations show that the dynamic information, captured by the deep motion features, seem to complement the static appearance information well. The final tracker achieves great results, outperforming state-of-the-art trackers, and yields a large improvement compared to the regular SRDCF. The best results were achieved when combining HOG, RGB features from a deep layer in an appearance CNN, and motion features from a deep layer in a CNN trained using optical flow images for action recognition.

An evaluation of the dimensionality reduction techniques PLS and PCA was also performed. The methods achieved very similar results. PLS got the highest score in terms of mean overlap precision and area-under-curve, and seems to handle larger dimensionality reductions better than PCA. However, the calculations take a bit longer, leading to PCA providing slightly better efficiency improvement. Employing dimensionality reduction on the proposed tracker using PLS improved the tracking results further, and decreased the time required for training by 57%.

As mentioned, the proposed tracker achieves excellent results when it comes to accuracy and robustness. However, as mentioned in the discussion of section 4.3, there are several factors that result in a lower frame count. Using faster CNNs, and looking into other ways of estimating the scale of the target are two ideas for future work that would increase the frame count.

The optical flow estimation method chosen is, as it turns out, not the fastest nor best method of choice. A possible choice could for example be [14], in which a CNN is constructed to solve the optical flow estimation problem, handling up to 10 frames per second with very good results. Employing such a method would make it possible to

integrate the optical flow calculations in the tracker, instead of pre-calculating offline.

The network used for the extraction of motion features is trained for action recognition, and may therefore not be optimal for the tracking task. Training a CNN for visual tracking purposes, using optical flow images as input, is something that could improve the tracking performance. Another interesting choice of future work is to train a joint CNN that can handle both RGB and optical flow images simultaneously.

Bibliography

[1] Vishnu Naresh Boddeti, Takeo Kanade, and B. V. K. Vijaya Kumar. Correlation filters for object alignment. In *CVPR*, 2013. Cited on page 8.

[2] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Yui M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. Cited on pages 7 and 8.

[3] T. Brox, A. Brugn, N. Papenberg, and J. Weickert. *High Accuracy Optical Flow Estimation Based on a Theory for Warping*. 2004. Cited on pages 14, 15, and 16.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. Cited on page 22.

[5] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. In *ICCV*, 2015. Cited on pages 16 and 22.

[6] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. Cited on page 14.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. Cited on page 12.

[8] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. Cited on pages 3, 8, 12, 27, and 39.

[9] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, 2014. Cited on pages 8, 13, and 39.

[10] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. Cited on pages 3, 4, 7, 8, 9, 10, 12, 23, 27, and 39.

[11] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshop*, 2015. Cited on pages 3, 9, 14, 16, and 39.

[12] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *CVPR*, 2016. Cited on page 39.

[13] S. de Jong. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263, March 1993. URL `http://dx.doi.org/10.1016/0169-7439(93)85002-X`. Cited on page 18.

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2758–2766, 2015. doi: 10.1109/ICCV.2015.316. URL `http://dx.doi.org/10.1109/ICCV.2015.316`. Cited on page 47.

[15] Michael Felsberg. Enhanced distribution field tracking using channel representations. In *ICCV Workshop*, 2013. Cited on pages 7 and 39.

[16] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9): 1627–1645, 2010. Cited on page 12.

[17] Joseph A. Fernandez, Vishnu Naresh Boddeti, Andres Rodriguez, and B. V. K. Vijaya Kumar. Zero-aliasing correlation filters for object recognition. *TPAMI*, 37(8): 1702–1715, 2015. Cited on page 9.

[18] Hamed Kiani Galoogahi, T. Sim, and S. Lucey. Multi-channel correlation filters. In *ICCV*, 2013. Cited on page 8.

[19] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation filters with limited boundaries. In *CVPR*, 2015. Cited on pages 9, 16, and 39.

[20] Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. Transfer learning based visual tracking with Gaussian process regression. In *ECCV*, 2014. Cited on pages 7 and 39.

[21] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. Cited on pages 16 and 22.

[22] Susanna Gladh, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Deep motion features for visual tracking. In *ICPR*, 2016. Cited on pages 4 and 45.

[23] Sam Hare, Amir Saffari, and Philip Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. Cited on pages 7, 8, and 39.

[24] Shengfeng He, Qingxiong Yang, Rynson Lau, Jiang Wang, and Ming-Hsuan Yang. Visual tracking via locality sensitive histograms. In *CVPR*, 2013. Cited on pages 7 and 39.

[25] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with

kernelized correlation filters. *PAMI*, 2015. doi: 10.1109/TPAMI.2014.2345390. Cited on pages 3, 8, 12, 27, and 39.

[26] Joao Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012. Cited on page 7.

[27] M. Hubert and K. Vanden Branden. Robust methods for partial least squares regression. *Journal of Chemometrics*, 17(10):537–549, 2003. ISSN 1099-128X. URL `http://dx.doi.org/10.1002/cem.822`. Cited on page 19.

[28] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012. Cited on pages 7 and 39.

[29] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, 2010. Cited on page 39.

[30] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojír, G. Nebehay, R. Pflugfelder, and G. Häger. The visual object tracking VOT2015 challenge results. In *ICCV workshop*, 2015. Cited on pages 26 and 27.

[31] Matej Kristan, Roman Pflugfelder, Ales Leonardis, Jiri Matas, and et al. The visual object tracking VOT 2014 challenge results. In *ECCV Workshop*, 2014. Cited on page 8.

[32] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. van den Hengel. *A Survey of Appearance Models in Visual Object Tracking*. *CoRR*, abs/1303.4803, 2013. Cited on page 1.

[33] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV Workshop*, 2014. Cited on pages 23 and 39.

[34] P. Liang, E. Blasch, and H. Ling. Object tracking benchmark. *TIP*, 2015. Cited on pages 26 and 27.

[35] Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *CVPR*, 2015. Cited on page 14.

[36] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. Cited on pages 3, 7, 8, 14, 16, and 39.

[37] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *CVPR*, 2015. Cited on page 39.

[38] Stephen Marsland. *Machine Learning, An Algoritmic Perspective*. CRC Press Inc, 2nd edition, 2014. Cited on pages 3, 16, and 17.

[39] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2016. URL `http://neuralnetworksanddeeplearning.com/chap6.html`. Cited on page 11.

[40] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. Cited on page 14.

[41] Shaul Oron, Aharon Bar-Hillel, Dan Levi, and Shai Avidan. Locally orderless tracking. In *CVPR*, 2012. Cited on page 12.

[42] Patrick Perez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *ECCV*, 2002. Cited on page 12.

[43] Horst Possegger, Thomas Mauthner, and Horst Bischof. In defense of color-based model-free tracking. In *CVPR*, 2015. Cited on page 39.

[44] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshop*, 2014. Cited on page 14.

[45] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *Subspace, Latent Structure and Feature Selection, Statistical and Optimization, Perspectives Workshop, SLSFS 2005, Bohinj, Slovenia, February 23-25, 2005, Revised Selected Papers*, pages 34–51, 2005. URL `http://dx.doi.org/10.1007/11752790_2`. Cited on page 18.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, April 2015. doi: 10.1007/s11263-015-0816-y. Cited on page 11.

[47] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S. Davis. Human detection using partial least squares analysis. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 24–31, 2009. doi: 10.1109/ICCV.2009.5459205. URL `http://dx.doi.org/10.1109/ICCV.2009.5459205`. Cited on pages 3 and 17.

[48] Laura Sevilla-Lara and Erik G. Learned-Miller. Distribution fields for tracking. In *CVPR*, 2012. Cited on page 39.

[49] Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014. URL `http://arxiv.org/abs/1404.1100`. Cited on page 17.

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. Cited on page 14.

[51] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. Cited on page 16.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. Cited on page 21.

[53] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human

actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. Cited on pages 16 and 22.

[54] J. van de Weijer, C. Schmid, Jakob J. Verbeek, and D. Larlus. Learning color names for real-world applications. *TIP*, 18(7):1512–1524, 2009. Cited on page 12.

[55] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. Cited on page 21.

[56] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013. Cited on page 8.

[57] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *PAMI*, 2015. Cited on pages 26 and 27.

[58] Jianming Zhang, Shugao Ma, and Stan Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014. Cited on pages 7, 8, 12, and 39.