UNIVERSITÉ DE MONTRÉAL

# VISUAL OBJECT TRACKING USING DEEP SIMILARITY NETWORKS

ZHENXI LI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2019

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

VISUAL OBJECT TRACKING USING DEEP SIMILARITY NETWORKS

présenté par: LI Zhenxi
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. DESMARAIS Michel, Ph. D., président
M. BILODEAU Guillaume-Alexandre, Ph. D., membre et directeur de recherche
M. BOUACHIR Wassim, Ph. D., membre et codirecteur de recherche
M. KADOURY Samuel, Ph. D., membre

## DEDICATION

*To my dear father and mother,*
*the best in the world...*

## ACKNOWLEDGEMENTS

I would first like to thank my research advisors, Prof. Guillaume-Alexandre Bilodeau, and Prof. Wassim Bouachir earnestly, for giving me the precious opportunity to do research under their guidance. It is their guidance that made me able to discover the unknown step by step.

I also sincerely appreciate The Fonds de recherche du Québec - Nature et technologies (FRQNT) and Mitacs, for providing me with research funding which helped me more focus on my research.

I also want to express my heartfelt thanks to my colleagues at LITIV lab and my friends at Polytechnique, thanking them for the pleasure of their company.

I want to thank my parents, they are always been there for me. It is their support that enables me to go further and further.

Finally, I sincerely thank members of the jury for taking time to review my thesis and give valuable comments.

# RÉSUMÉ

Le suivi d'objets est une des tâches fondamentales de la vision par ordinateur. Étant donné une cible initiale dans la première trame, l'objectif du suivi est de trouver la nouvelle position de la cible dans les trames subséquentes. Les filtres de corrélation est une technique largement utilisée dans les applications de reconnaissance de formes pour mettre en correspondance des objets. Ainsi, de nombreuses méthodes de suivi sont basées sur les filtres de corrélation. Bien que ces méthodes de suivi atteignent un bon compromis entre la précision et la vitesse de suivi, leurs performances sont limitées par l'utilisation de caractéristiques d'apparence définies manuellement. Récemment, les méthodes d'apprentissage profond ont démontré des résultats impressionnants dans diverses tâches de vision par ordinateur. De ce fait, un grand nombre de méthodes de suivi basées sur l'apprentissage profond ont été proposées pour améliorer la robustesse et la capacité à discriminer les objets durant le suivi. Cependant, la représentation des caractéristiques de la cible par un seul réseau neuronal convolutif (RNC) n'est souvent pas assez discriminante. De plus, l'architecture du RNC doit être choisie avec soins pour réduire le temps de calcul lors du suivi. Dans notre travail, nous avons exploré différentes méthodes pour améliorer le pouvoir discriminant d'un RNC pour le suivi d'un objet. Nous avons proposé deux méthodes de suivi, MBST (*Multi-Branch Siamese Tracker*) et MFST (*Multiple Features-Siamese Tracker*). Les deux méthodes sont basées sur des RNC avec une architecture siamoise. Cette architecture définit le suivi comme un problème d'apprentissage par similarité.

L'algorithme MBST utilise plusieurs branches convolutives pour extraire des représentations diverses sur l'objet à suivre, notamment des branches dépendantes du contexte et une branche sémantique. Les branches dépendantes du contexte sont entrainées pour représenter plusieurs catégories spécifiques d'apparence d'objets, codant ainsi plus d'informations de contexte pour le suivi. La branche sémantique est entrainée pour résoudre la tâche de classification des images, ce qui permet de mieux utiliser les informations sémantiques, c'est-à-dire les caractéristiques propres à une catégorie d'objets. Afin de mieux utiliser ces branches, nous avons proposé un mécanisme de sélection en ligne, qui sélectionne de manière dynamique la meilleure branche en fonction du pouvoir de discrimination calculé à partir de cartes de corrélation. Basée sur une architecture multibranche et un mécanisme de sélection en ligne, la méthode MBST obtient des performances supérieures par rapport aux méthodes standards basées sur l'architecture siamoise.

Observant que les caractéristiques des sorties provenant de différentes couches de convolution

d'un RNC intègrent différents niveaux d'abstraction de la cible, et que les différents canaux des cartes de caractéristiques du RNC jouent différents rôles dans le suivi, nous avons proposé la méthode MFST. Pour intégrer les différents niveaux d'abstraction, notre méthode utilise de façon hiérarchique des cartes de caractéristiques de convolution issues de deux modèles de réseaux neuronaux convolutifs. Ces cartes de caractéristiques sont aussi recalibrées pour utiliser les canaux les plus appropriés pour décrire l'objet. Nous proposons également une nouvelle stratégie de combinaison de caractéristiques afin de fusionner les diverses représentations de la cible, ce qui génère un modèle d'apparence plus riche et robuste. Basée sur une représentation plus riche de la cible, la méthode MFST permet d'obtenir une précision de suivi plus élevée et surpassent plusieurs méthodes récentes de l'état de l'art.

# ABSTRACT

In this thesis, we study visual object tracking using deep similarity networks. Visual object tracking is a fundamental task in computer vision. Many approaches based on correlation filters and deep learning have been proposed to solve this problem. Inspired by deep learning-based methods, we exploit the siamese network model to address object tracking, by formulating tracking as a similarity learning problem.

Most deep learning methods only use features extracted from the last convolutional layer of a single model which lead their tracker to be prone to fail when target appearance changes significantly. Different features extracted from different convolutional models encode the target object in different ways. Trackers with diverse feature representations can be adapted more easily to challenging scenarios. Besides, the discriminative power of features from the last convolutional layer is very limited. Since convolutional features from different layers contain different levels of abstraction of the target object, discovering an appropriate scheme to fuse hierarchical features is also beneficial for tracking.

Based on these observations, we proposed two trackers, MBST (Multi-Branch Siamese Tracker) and MFST (Multiple Features-Siamese Tracker). Both trackers are built on the siamese architecture, addressing tracking as a similarity learning problem. The Multi-Branch Siamese Tracker employs multiple convolutional models to extract diverse feature representations for the target object. It consists of context-aware branches, which are trained to track for several specific appearance categories, and a semantic branch, which is trained to solve image classification task and is used to extract semantic information and diversify feature representation. To make better use of each branch, we propose an online branch selection mechanism to dynamically select the optimal branch according to their discriminative power. The second tracker Multiple Features-Siamese Tracker utilizes convolutional features from multiple layers of two convolutional models with a feature recalibration mechanism. With diverse features from different layers and different models, the combined representation embeds not only the high-level abstraction of the target object but also its detailed representation. We also propose a feature recalibration mechanism to apply channel-wise weights on the feature maps, in order to enhance discriminative channels. Both of these two trackers demonstrate improved performance compared to standard Siamese trackers.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ACRONYMS

| | |
|---|---|
| AUC | Area Under Curve |
| CN | Color Name |
| CNN | Convolutional Neural Network |
| DCF | Discriminative Correlation Filter |
| DFT | Discrete Fourier Transform |
| FFT | Fast Fourier Transform |
| FPS | Frame Per Second |
| HOG | Histogram of Oriented Gradients |
| IoU | Intersection over Union |
| KCF | Kernelized Correlation Filter |
| MBST | Multi-Branch Siamese Tracker |
| MFST | Multiple Features-Siamese Tracker |
| PNR | Peak-versus-Noise Ratio |
| ReLU | Rectified Linear Units |
| SE | Squeeze-and-Excitation |
| SSE | Sum of Squared Error |
| VOT | Visual Object Tracking |

# LIST OF APPENDICES

# CHAPTER 1 INTRODUCTION

As a fundamental task in computer vision, visual object tracking plays an important role in many modern applications. For example, in autonomous driving, sensors must track surrounding obstacles to estimate how those obstacles are moving and predict their trajectories in the future. With the help of tracking technology, autonomous driving gets much safer. In video surveillance, different type of targets need to be monitored. Once a target is labeled, the tracking algorithm focuses on it. Based on this, the surveillance system is able to keep watch on the target and analyses where the target is and what the target is doing. Besides, many face beauty camera applications locate and track users' face using such technologies. With the known face area, these applications focus on the face and edit this area to obtain better-looking pictures. Furthermore, with visual object tracking technology, unmanned aerial vehicles get much more intelligent. Once the target is assigned, they track and follow targets to perform more tasks.

## 1.1 Motivation

Despite significant progress in recent decades, large appearance changes that arise from occlusion, deformation, illumination variation, abrupt motion, and background clutter are still challenging for visual object tracking. These obstacles result in an error-prone target representation. We summarize them as three main problems, 1) object appearance variations, 2) appearance model update and drift and 3) target scale variation.

**Object appearance variations** In the problem of object appearance variations, there are two vital appearance changes that cause significant difficulties. The first one is, for different tracking tasks, the target objects may be different. The target objects we need to track can be a car, a face, a human body or anything else assigned. Different objects have different appearance which leads to varied input for the object tracking algorithm. The other one is that after we know the target object, there are still some unstable visual attributes in the frame sequences as illustrated in Table 1.1.

All these unstable factors may happen in any video sequence in the tracking scenario. Therefore, the feature representation used in tracking should be robust enough to handle these challenges.

Table 1.1 Unstable attributes in the frame sequences in visual object tracking. (This table is adapted [reprinted] from [1] © 2015 IEEE)

| Attr | Description |
|---|---|
| IV | Illumination Variation—The illumination in the target region is significantly changed |
| SV | Scale Variation—The size of the object changes over different frames |
| OCC | Occlusion—The target is partially or fully occluded |
| DEF | Deformation—Non-rigid object deformation |
| MB | Motion Blur—The target region is blurred due to the motion of target or camera |
| FM | Fast Motion—The motion of the target is large |
| IPR | In-Plane Rotation—The target rotates in the image plane |
| OPR | Out-of-Plane Rotation—The target rotates out of the image plane |
| OV | Out of-View—Some portion of the target leaves the view |
| BC | Background Clutter—The background near the target has the similar color or texture as the target |
| LR | Low Resolution—The target region has only a few pixels |

**Appearance model update and drift** Either in discriminative correlation filter-based (DCF) tracking (Section 2.3) or in convolutional neural networks-based (CNNs) tracking (Section 2.4), most of tracking approaches keep and update an appearance model for tracking. Using the appearance model, the target is detected in video sequences. In many trackers [5, 16, 17], the image patch assigned in the first frame is used to train the appearance model. This image patch not only contains the target but also includes some context to be more discriminative. After the target position is detected in the next frame, the appearance model is updated at the new position. Thus, these tracking approaches use an appearance model updating strategy that detects and updates for every frame. If a tracking error happens, the appearance model drifts away from the visual appearance of the target.

Rather than updating the model every frame, ECO [18] uses a sparser updating scheme which is commonly used in non-DCF trackers. The sparser updating scheme updates the model at a fixed interval. However, an optimal updating mechanism should update the model once sufficient changes in the target have occurred and when the target is tracked correctly. In ECO [18], the model is updated at a fixed interval $N_s$. The model is optimized every $N_s$ frames. When $N_s$ is 1, the optimization process would be performed in every frame, as in standard DCF trackers. Through experiments, they found an empirical interval $N_s \approx 5$, which achieved better tracking performance on average.

Regardless of updating the model every frame or at a fixed interval, when the tracking is inaccurate, the output result may introduce negative background information to the appearance model. In the worst case, the appearance model may stop representing the target object since the target object gets lost gradually, which is called model drift.

To solve this problem, another model updating strategy is proposed in [3]. The model updating is determined by two criterions which are based on the value of the peak-versus-noise ratio (PNR) and the maximum value of the response map. Ideally, the response map generated by DCF trackers should only contain one peak value which should correspond to the ground truth position of the target object. However, the response may fluctuate and contain more peak values when the target is occluded as shown in Figure 1.1. In the meantime, the PNR value decreases significantly. Thus, considering the peak value of the response map and the PNR value at the same time can avoid unexpected updating when the occlusion happens. Otherwise, negative background information will be introduced into the appearance model which lead to model drifting gradually.

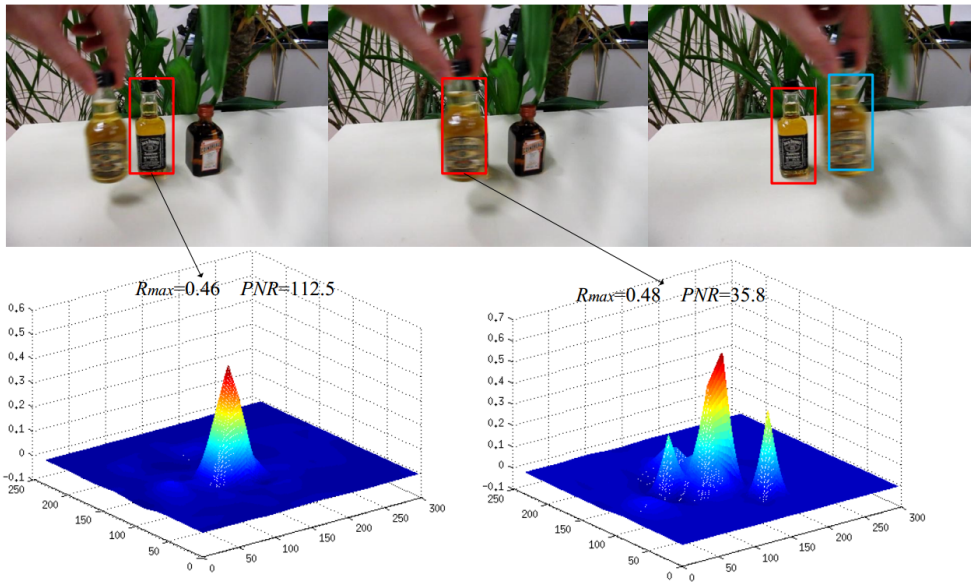



Figure 1.1 The model update strategy proposed in [3]. The target object is occluded in the second frame, which lead to more peak values in the corresponding response map. In the corresponding map, although $\mathcal{R}_{max}$ remains large, the PNR value significantly decreases. Considering the two constraints, the unexpected model updating is avoided. (This image is adapted [reprinted] from [3] © 2017 IEEE)

**Target scale variations** Generally, the target scale changes significantly during tracking which is not only caused by the target itself, but can also result from camera's movement.

The size of target can get larger or smaller at any time. However, the tracking system is only given the initial bounding box in the first frame. If the tracking system keeps using a bounding box with the same size to locate the target, the labeled area located by the tracking system could contain many other neighboring areas and the tracking accuracy will decrease a lot. Thus, another challenge is to learn to adapt the bounding box against target scale variations.

The key to solving these problems is to find expressive features and corresponding trackers that maintains a robust appearance model. The feature representation should allow the tracker to differentiate the target object from the deceptive background and also tolerate the appearance changes of the target object. Therefore, the goal of our work is to explore different ways to represent the target object. Besides, our work also includes finding a robust model update strategy to deal with the drifting problem and target scale variation.

## 1.2 Basic Concepts

To obtain a robust and effective feature representation, some works [5, 19] use correlation filters to model the appearance of objects. With the kernel trick employed in these work [5, 19, 20], these methods achieve almost real-time speed while keeping high accuracy. At the same time, as the neural networks are getting more and more popular in the last decade, they show outstanding performance on many computer vision tasks. Convolutional neural network models have also been used in many approaches [7, 8, 21, 22] to obtain robust object appearance description. To understand visual object tracking more clearly, we introduce some basic concepts in this section including kernelized correlation filters, convolution neural networks (CNN), siamese networks and some frequently-used deep convolutional neural network models.

### 1.2.1 Kernelized Correlation Filter

The Kernelized Correlation Filter (KCF) is a well-known single target tracking method, which has been used in many works [5, 19, 20]. Due to the fact that training examples can be represented as a circulant matrix, correlation filter methods are very efficient.

Correlation with an example template is a common way to detect patterns in images. Correlation filter-based trackers first model the appearance of target using filters trained on the template image. Then they detect the object in the next frame by correlation with the filter inside a search window. The position of maximum value in the output response indicates the position of the target. An online update is performed based on the detection result.

Considering that there is a large number of samples to train, a circulant matrix is used to store samples since circulant matrix can be diagonalized by Discrete Fourier Transform (DFT). The off-diagonal elements of the circulant matrix are zero while the diagonal elements represent eigenvalues which are equal to the DFT transformation of the sample ($x$). Therefore, the KCF employs kernel to $x$ transforming data to a feature domain. The kernelized $x$ is shifted and forms the circulant matrix. Such kernelization operation lead to $\mathcal{O}(n\log(n))$ complexity rather than $\mathcal{O}(n^2)$ or even higher complexities in other kernel algorithms. Thus, the process can be formulated as a ridge regression problem:

$$E_h = \min_{\boldsymbol{h}} \frac{1}{2} \parallel \boldsymbol{y} - \sum_{c=1}^{C} \boldsymbol{h_c} * \boldsymbol{x_c}||^2 + \frac{\lambda}{2} \sum_{c=1}^{C} ||\boldsymbol{h_c}||^2 \tag{1.1}$$

where the objective is to minimize $E_h$ given the training sample $x$ and the desired response $y$, $h$ is the learned filter. More details are presented in [19].

### 1.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a class of feedforward neural networks containing convolutional calculations with a deep structure. Generally, the CNN architecture consists of an input layer, output layer, and multiple hidden layers. The hidden layers contain convolutional layers, pooling layers, activation layers, fully connected layers, and normalization layers.

The convolution operation includes two functions of a real-value argument. With $x$ as an input, the convolution operation is denoted as:

$$s(t) = (x * w)(t) \tag{1.2}$$

where the function $w$ is referred to as the kernel, the output is typically referred to as the feature map. An example of 2D convolution is shown as Figure 1.2.

### 1.2.3 Deep CNN Models

Since the pioneering work of AlexNet [23] that proposed to address the ImageNet [24] classification task, a large number of follow-up deep CNN models [2, 25–27] were proposed. Many experiments show that deep CNNs are able to handle most kinds of computer vision tasks very well. Here we present two deep CNN models that are frequently used in object tracking.

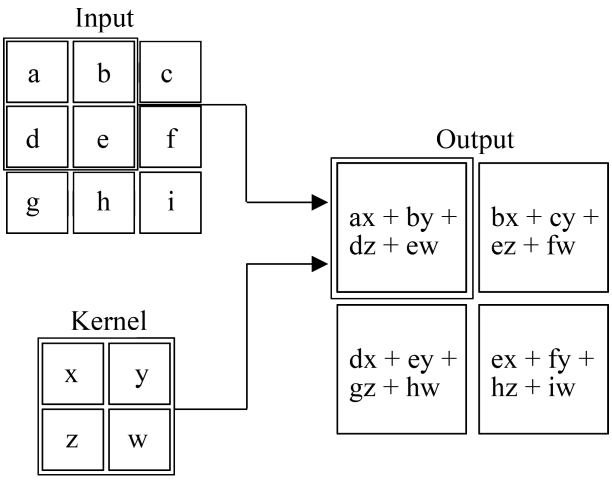



Figure 1.2 An example of 2D convolution without kernel flipping.

**AlexNet** The original AlexNet [23] was proposed to solve a classification task, which classifies high-resolution images into 1000 classes. It consists of five convolutional layers, three fully-connected layers and a number of max-pooling layers following the convolutional layers. The final classification result is produced by a 1000-way softmax. Due to its efficient feature representation ability, many works take the first five convolutional layers of AlexNet to extract feature representations while ignoring the fully-connected layers. The architecture of AlexNet [23] is shown as Figure 1.3.

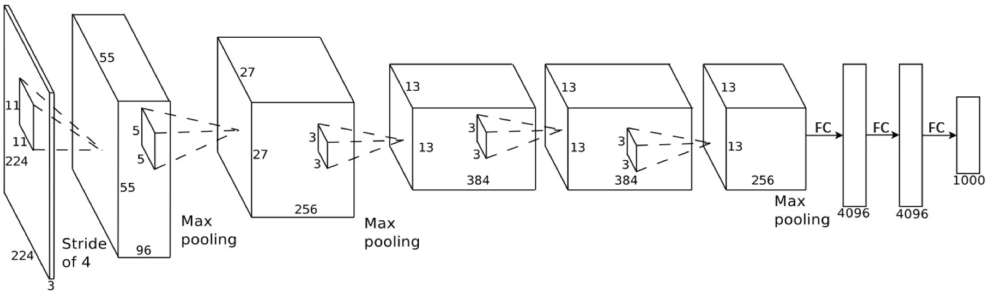



Figure 1.3 An illustration of the architecture of AlexNet.

**VGG** To investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting, Simonyan et al. [2] proposed an architecture with very small ($3 \times 3$) convolution filters, VGG networks. The proposed VGG networks show a significant improvement and pushes the depth of networks to 16-19 weight layers as shown in Table 1.2. It is remarkable that by stacking $3 \times 3$ conv. filter layers, $7 \times 7$ and $5 \times 5$ conv.

filter layers can be superseded, and fewer parameters are required. Besides, more non-linear layers can be incorporated, which makes the decision function more discriminative.

Table 1.2 The network configuration of VGG [2] networks. (This table is adapted [reprinted] from [2] © 2015 ICLR)

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input($224 \times 224$ RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

### 1.2.4 Siamese Networks

Nowadays, many deep learning-based trackers use siamese networks as basic framework. The siamese networks are a special type of network structure. It consists of two identical neural networks, each taking one of two input images. The last layers of the two networks are then fed to the loss function. The structure of the siamese network is as shown in Figure 1.4. Intuitively, we can put the exemplar image and the search image into the network to extract feature representations for them. Taking the feature representation of the exemplar image as a filter, a cross-correlation can be performed on the feature representation of the search image to generate a response map. Each value of pixels in the response map indicates the possibility to be the new location of the target. To train the network, the loss is defined to be the mean of the individual difference between the response map and a ground truth map.

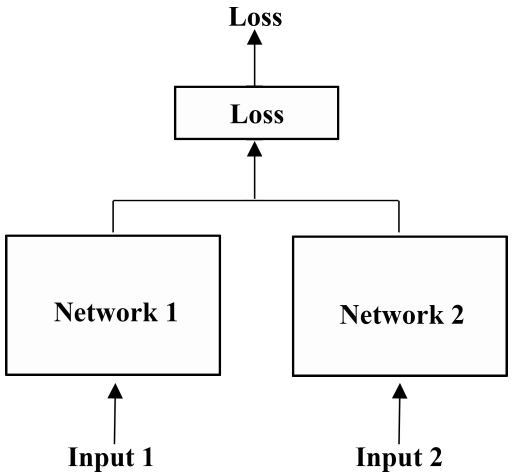



Figure 1.4 The structure of siamese networks. It contains two identical networks, each of them takes one of the inputs and their outputs are fed to the loss function.

## 1.3 Research Objectives

The main objective of this thesis is to discover a robust and efficient feature representation for visual object tracking, which is able to deal with appearance variations. Furthermore, our work aims at solving the model drifting problem and scale variation problems. Therefore, the research objectives of this work can be summarized as follows:

- To discover a robust and efficient feature representation for visual object tracking, which can handle appearance variations.

- To investigate the combination of various features which can make full use of feature representation extracted by our system.
- To develop a model update strategy that addresses the model drifting problem and is aware of object scale variations.

### 1.4 Thesis Outline

Our thesis is organized as follows. We first introduce some related work and the evaluation metrics in Chapter 2. Then, we give an overview of the approaches we proposed in Chapter 3. Our approaches are described in Chapter 4 and Chapter 5. Chapter 6 presents general discussion and Chapter 7 concludes the thesis.

# CHAPTER 2 LITERATURE REVIEW

In this chapter, we define and discuss the visual object tracking (VOT) problem and some related concepts. We also introduce the main state-of-the-art approaches in VOT.

## 2.1 Visual Object Tracking

Visual object tracking (VOT) can be defined as the process of locating a region of interest in a video sequence, which contains four steps: target initialization, appearance modeling, motion prediction, and target positioning [28]. The first step, target initialization, is to annotate the target object with any of the following representations: centroid, ellipse, object bounding box, object contour or object skeleton. Usually, the target object is annotated in the first frame of the video sequence with a bounding box, and the trackers locate the target object in the following frames.

Appearance modeling consists of extracting visual features for a better representation of the target object, and in constructing mathematical models which are used for object detection with learning approaches. Then, the position of the target is predicted in the motion prediction step for subsequent frames. The target positioning step outputs the location of the target by greedy search or maximum posterior prediction. The VOT process is challenging due to many real-world difficulties:

- loss of information in image projection;
- low image quality;
- uncertainties of target object;
- movements of target object;
- illumination variation;
- object occlusion;
- real-time tracking constraints.

Tracking problems can be simplified by applying constraints on the appearance and motion models. For example, we can assume that there are no significant appearance changes in sequential frames and the motion of the target object is smooth. Furthermore, we can also

assume that the velocity of the moving target is constant or accelerates constantly. The visual object tracking process is illustrated as Figure 2.1.

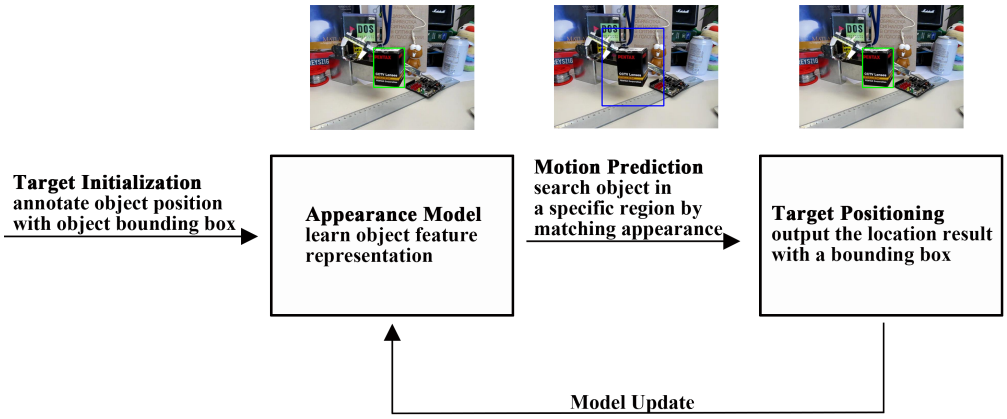



Figure 2.1 The illustration of visual object tracking process. It includes target initialization, appearance model, motion prediction and target positioning. Appearance model is updated by new target appearance.

Several works have been conducted on object tracking. The main differences between these works come from different object representation, image features and the model they used. To represent the tracked object, three types of methods can be used, hand-crafted features, correlation filters, and deep features. Our review focuses on different object representations and two types of methods are described in Section 2.3 and Section 2.4.

## 2.2 Feature Descriptors

To distinguish between the target and the surrounding environment, different feature descriptors can be used to represent them. In this section, we introduce three popular handcrafted feature descriptors: Color Histogram, Color Name (CN) and Histogram of Oriented Gradients (HOG), which are the most frequently used in visual object tracking in the last few years for methods that are not based on deep learning.

**Color Histogram and Color Name** Color is one of the important characteristics of our world. As such, it is also a significant feature in computer vision to describe images and objects. However, its description is complicated since there are lots of uncertainties in many tasks. To understand visual data better, it is crucial to make better use of color information.

Color histogram is a summarization of the color distribution in an image, it can be built from images in various color spaces, like RGB or HSV. To collect the color distribution, we

can divide the color values into several bins. The number of pixels in each bin will be used as the value of the corresponding histogram. The color histogram represents the statistics of different types of color, which is typically used in appearance modeling in visual object tracking.

Color names (CN), which are linguistic color labels representing colors, showed excellent results for computer vision tasks. The work [29] concludes that there are eleven basic color terms: red, orange, yellow, green, blue, purple, grey, black, pink, white and brown. Thus, color naming is associating RGB observations with color labels. Weijier et al. [30] proposed an approach mapping RGB values with color names, which are automatically learned from real-world image. Danelljan et al. [31] investigated the color contribution in a tracking-by-detection framework with a color mapping. The adaptive color names-based approach shows superior performance for object tracking.

**Histogram of Oriented Gradients** In computer vision, the histogram of oriented gradients (HOG) is a texture descriptor based on shape and edge, which can be used to detect objects. The basic idea of HOG is to use gradient information to reflect the edge information of the target image and to characterize the local appearance and shape of the image through the value of the local gradient. Bhat et al. [15] used HOG features and other features in object tracking and obtained good results.

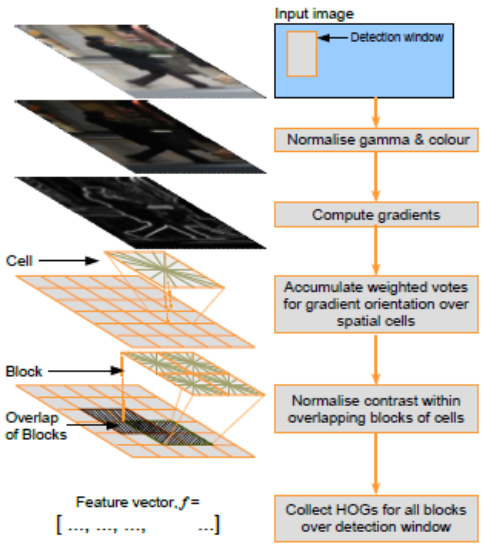



Figure 2.2 Illustration of HOG descriptor. (This image is adapted [reprinted] from [4] © 2014 ELSEVIER)

HOG feature extraction can be represented by the process shown in Figure 2.2: (1) normalize

the color space to reduce the influence of illumination and background, (2) divide the detection windows into cells of the same size and extract corresponding gradient information, (3) combine adjacent cells into large overlapping blocks, which can effectively utilize overlapping edges information and be used to build the histogram statistics of the whole block, (4) the gradient histogram in each block is normalized to further reduce the influence of background and noise, (5) the HOG features of all blocks in the whole window are collected and the feature vectors are used to represent the features. In this process, the parameters template of different scale, the selection of gradient direction, the size of overlapping blocks and cells, as well as the normalization factor, will affect the final results.

### 2.3 Tracking based on Correlation filter

Correlation is a common measure in pattern recognition. In spite of much known weakness, its efficiency and simplicity have attracted continuous research on it. A common way to use correlation is the correlation filter-based tracking approaches. Given the target object in the first frame, the correlation filter-based trackers train filters on the specific region to encode the target appearance. From this point on, the trackers detect the target object in the subsequent frames and update the filters based on the newly detected object. The tracking process starts from correlating the filter over a sub-window in the next frame and generate a response map. The value of each pixel indicates the possibility of being the new position of the target. Thus, the location of the maximum value is regarded as the result. The filter is updated on the new location then.

To speed up tracking, correlation is employed in the Fourier domain with the Fast Fourier Transform (FFT). First, the input image $f$ and the filter $h$ are transformed:

$$\begin{aligned} F &= \mathcal{F}(f) \\ H &= \mathcal{F}(h). \end{aligned} \tag{2.1}$$

It is known that correlation is an element-wise multiplication in the Fourier domain, which can be a form of:

$$G = F \odot H^* \tag{2.2}$$

where $\odot$ denotes the element-wise multiplication operation and $*$ indicates the complex conjugate. Afterward, the output $G$ is transformed back into the spatial domain by the inverse FFT. With the use of FFT, correlation filters-based approaches obtain high computational efficiency. The performance of correlation filters has been greatly extended in object tracking. The following subsections discuss in detail about kernelized correlation filters and MOSSE [16]

filters.

### 2.3.1 Kernelized Correlation Filters

Most correlation filter trackers are discriminative classifiers, whose purpose are to classify the target and surrounding environment. Considering that these trackers are trained with translated and scaled sample patches, Henriques et al. [5] proposed kernelized correlation filters which shift training samples into circulant matrix and use kernelized regression for detection.

**Cyclically shifted samples** Since discriminative classifiers are trained online with samples collected during tracking, the tracking speed is seriously affected by the potentially large number of samples. Whereas, limiting the training samples may be at the cost of discrimination performance. Henriques et al. [32] observed that this problem can be solved by using circulant structure.

The circulant structure consists of a base sample and several virtual samples by translating the base sample. Regarding an $n \times 1$ vector $\boldsymbol{v}$ as the target patch, using a *cyclic shift operator* can model 1D translations:

$$C = \begin{bmatrix} 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{2.3}$$

The product $C(\boldsymbol{v}) = [\boldsymbol{v}_n, \boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_{n-1}]^T$ shifts $\boldsymbol{v}$ by one element, modeling a small translation. A larger translation can be obtained by chaining $u$ shifts as $C^u\boldsymbol{v}$. A negative $u$ shifts in the reverse direction. Examples are shown in Figure 2.3 and Figure 2.4.

The circulant structure used can encode the convolution of vectors, which is close to what we do when evaluating a classifier at many different subwindows. Due to the properties of circulant matrices, they can be computed in the Fourier domain, which makes the learning extremely fast while the detection still performs well.

**Ridge regression with shifted samples** Since the ridge regression admits a close-form solution and can achieve outstanding performance, it can be used to train correlation filters. The objective of training is to find a correlation function $f(\boldsymbol{v}) = w^T\boldsymbol{v}$ that minimizes the

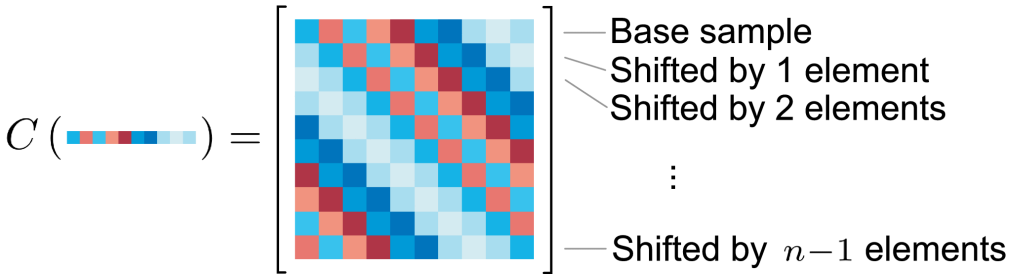



Figure 2.3 An example of translating 1D signal to generate a circular matrix. The rows are cyclic shifts of the base sample. (This image is adapted [reprinted] from [5] © 2015 IEEE)

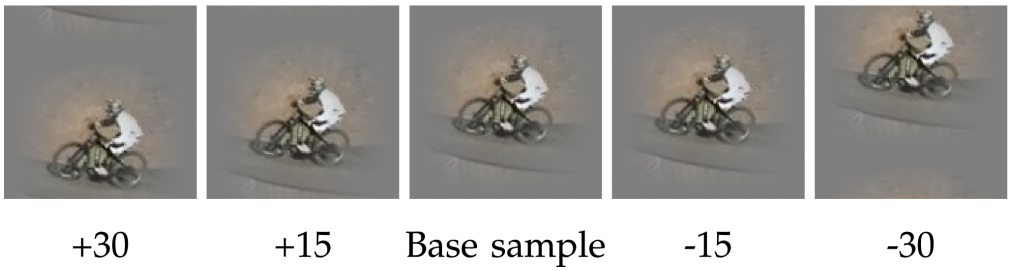



Figure 2.4 Illustration of vertical cyclic shifts of a base sample. (This image is adapted [reprinted] from [5] © 2015 IEEE)

Euclidean distance between the samples $\boldsymbol{v}$ and the expected outputs $y$ with a regularization term:

$$\min_{w} \sum (f(\boldsymbol{v}) - y)^2 + \lambda ||w||^2 \tag{2.4}$$

To solve the regression problem, all possible cyclic shifts can be used as a circulant matrix $\boldsymbol{V}$:

$$\boldsymbol{V} = C(\boldsymbol{v}) = \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_n \\ v_n & v_1 & v_2 & \dots & v_{n-1} \\ v_{n-1} & v_n & v_1 & \dots & v_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_2 & v_3 & v_4 & \dots & v_1 \end{bmatrix} \tag{2.5}$$

The first row is the base sample $\boldsymbol{v}$, then each row is shifted one element to the right as Figure 2.3 illustrates. With the circulant matrix, the training process is much more efficient by diagonalizing. Besides, the "kernel trick" can be used to advance the training. Since the optimization problem is linear, the complexity of evaluating the regression function is linearly associated with the number of samples. However, with kernel trick, the inputs of a linear problem can be mapped to a non-linear feature-space.

### 2.3.2 MOSSE Filters

The core idea of Minimum Output Sum of Squared Error (MOSSE) [16] is to produce correlation filters with fewer training images. Taking a set of training images $f_i$ and expected outputs $g_i$, typically, $g_i$ is generated from ground truth which has a compact 2D Gaussian shaped peak in the training image $f_i$ centered on the target object. By applying the training in the Fourier domain, a simple element-wise relationship can be built for the input and the output as shown in Equation 2.6.

$$H_i^* = \frac{G_i}{F_i} \tag{2.6}$$

where $F_i, G_i$ are the Fourier transform of $f_i$ and $g_i$, $H$ is the correlation filter to be trained.

To find the optimal filter that maps the training inputs to the expected outputs, MOOSE proceeds by minimizing the Sum of Squared Errors (SSE) to solve this problem, as shown in Equation 2.7.

$$\min_{H^*} \sum_i |F_i \cdot H^* - G_i|^2 \tag{2.7}$$

The idea of using SSE to find the correlation filter has been used many correlation filters-based tracking methods. Whereas, rather than assuming that the target was always centered in the input and the output was fixed. MOSSE customize every output $g_i$. Since the target is not always centered in the tracking problem, the Gaussian shaped peak moves following the target in the input, the idea of customizing every output $g_i$ is more reasonable.

## 2.4 Tracking based on CNNs

Nowadays, many computer vision tasks have benefited from convolutional neural networks (CNNs), such as image classification, object detection, semantic segmentation, and many others. Visual object tracking also takes advantage of CNNs due to its feature representation ability. In this section, we introduce CNN-based tracking approaches, including tracking by regression networks, tracking by siamese networks, tracking with the combination of representation learning and correlation filter and tracking with visual attention.

### 2.4.1 Tracking by regression networks

Li et al. [6] proposed DeepTrack, using a complex Convolutional Neural Network model for learning effective feature representations of multiple input cues. The CNN model consists of two convolutional layers and two fully-connected layers. The ReLU (Rectified Linear Unit) is adopted as the activation function, and max-pooling operators are used for dimension-

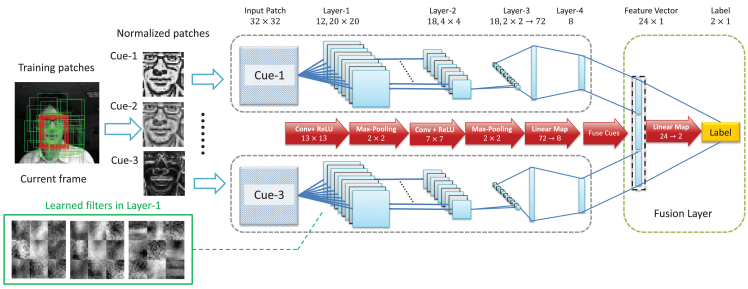



Figure 2.5 The CNN architecture of DeepTrack with multiple image cues. The gray dashed blocks are the independent CNN channels for different image cues, the green dashed block is the fusion layer where a linear mapping is learned. (This image is adapted [reprinted] from [6] © 2016 IEEE)

reduction. They use locally normalized image patches as input, which draws a balance between the representation power and computational load. After convolutional layers, a 72-dimensional feature vector is generated. Then, fully connected layers map the 72-D feature vector into a 2-D confidence vector which corresponds to the positive score and negative score for the image patch. The CNN architecture of DeepTrack is shown in Figure 2.5.

Even though DeepTrack [6] has achieved remarkable performance improvement over correlation filter-based trackers, the tracking speed of those trackers [6,33] trained online are limited. Held et al. [7] proposed a regression network-based method GOTURN, which is trained offline, tracking objects at 100 fps. Without online training, the GOTURN tracker learns a generic relationship between object motion and appearance. The network architecture of GOTURN is shown in Figure 2.6.

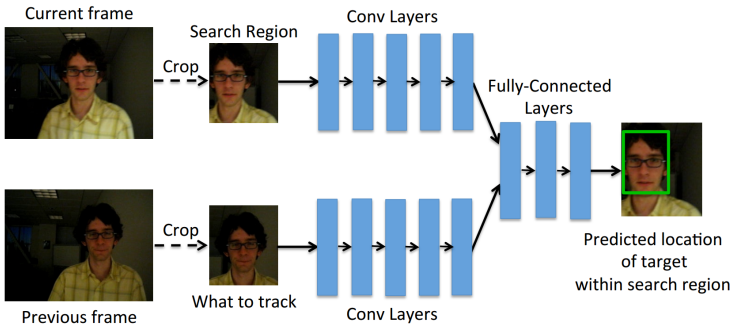



Figure 2.6 The network architecture of GOTURN [7]. It first extracts features of target object and searches region by a set of convolutional layers. Features are then fed through the fully connected layers to estimate the movement of the target object. (This image is adapted [reprinted] from [7] © 2016 Springer)

Using regression networks, GOTURN [7] formulates the tracking problem as an image com-

parison problem. It employs a sequence of convolutional layers to extract the deep feature of the target object and the search region. Then, the outputs are fed through three fully connected layers. The fully connected layers compare the features of the target object and the features of the search region in the current frame to find the movement of the target object. Since the networks are fully trained offline on a collection of videos and no online training is needed in tracking, it tracks novel objects in a fast and accurate and robust manner.

### 2.4.2 Tracking by siamese networks

Trackers based on regression networks [6,7,33] predict the bounding box of the target object directly. Unlike regression networks-based trackers [6,7,33], siamese networks-based trackers [8,34,35] address tracking as a similarity learning problem. The pioneering work, SiamFC [8], takes two identical branches to extract deep features from the exemplar image and the search image. The identical feature extraction branches are fully convolutional layers. After that, the feature map extracted from the exemplar image is regarded as the filter to perform correlation on the feature map extracted from the search image. By computing the similarity of all translated sub-window effectively, the position of maximum value in the output response map is the location of the target object. The architecture of SiamFC [8] is as shown in Figure 2.7.

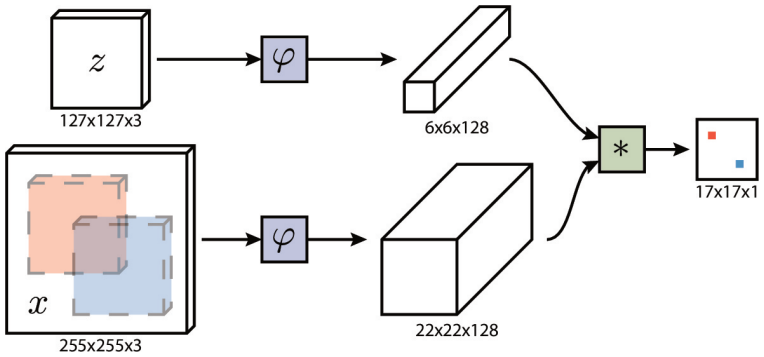



Figure 2.7 The architecture of SiamFC [8]. The inputs are the exemplar image and the search image. Then, two identical branches made up of fully convolutional neural networks, extract feature representations for inputs. The output is a score map produced by correlation using feature representation of exemplar image on search image. In this example, the value of red and blue pixels indicate the similarities for the corresponding sub-windows. (This image is adapted [reprinted] from [8] © 2016 Springer)

### 2.4.3 Representation learning for correlation filter

Correlation Filter trains a linear template to classify the target and surrounding environment. These correlation filter based trackers [5, 19, 20] adopt features that are either manually designed or trained for a different task. While on the other hand, deep features which are learned from data show robust representation to describe objects. SiamFC [8] focuses on learning a similarity function to discriminate whether two images patches contain the same object or not, by exploiting deep features. However, a fixed metric learned offline prevents the tracker from utilizing any video-specifies cues which may be helpful for discrimination. In light of these limitations, Valmadre et al. [9] incorporates a Correlation Filter into a siamese network. Instead of using the convolutional feature of the target object directly, they employ a correlation filter learner in the network which enables learning features coupled to the correlation filter. The architecture of CFNet is shown as Figure 2.8.

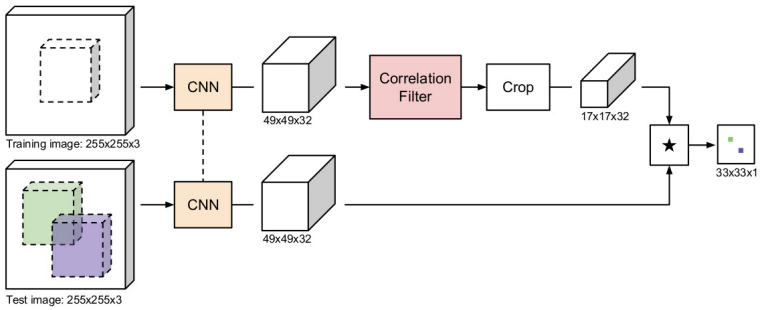



Figure 2.8 The network architecture of CFNet. The first part is two identical branches made up of fully convolutional neural networks like SiamFC [8]. The difference between CFNet [9] and SiamFC [8] is that CFNet employs an additional correlation filter learner layer to learn filter on the feature representation of the exemplar image. (This image is adapted [reprinted] from [9] © 2017 IEEE)

After incorporating a correlation filter layer into the similarity network during training, it enables shallow networks to rival their slower, deeper counterparts. CFNet [9] can achieve the same tracking accuracy as the basic siamese tracker SiamFC [8].

CFNet [9] learns correlation filter on the feature from the last convolutional layer. Different from CFNet [9], Ma et al. [21] utilize features from different convolution layers to learn the correlation filters. They observed that features from different levels of convolution layers contain different information, which is complementary. The last layers of CNNs are efficient to capture semantics. However, they are insufficient for capturing details. On the contrary, the earlier layers, are precise in localization since more details are kept, while they are unable to capture semantics as illustrated in Figure 2.9. Based on this observation, they adaptively

learn correlation filters on each convolution layer to encode target appearance.

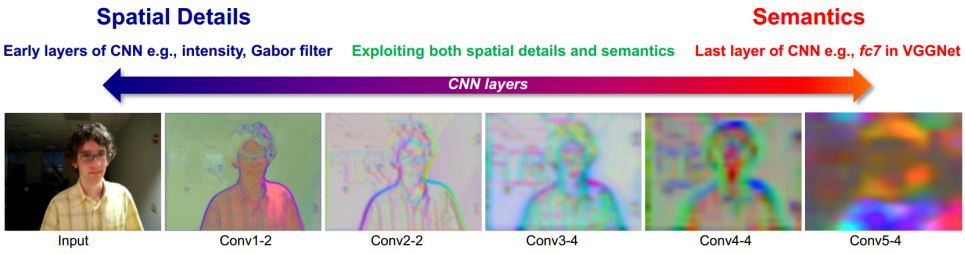



Figure 2.9 Visualization of deep hierarchical feature. Features from earlier layers contain more low-level information similar to the response map of Gabor filters [10]. On the contrary, features from deeper layers capture more semantic information and less spatial details. (This image is adapted [reprinted] from [11] © 2018 IEEE)

### 2.4.4 Tracking with Visual Attention

The visual attention scheme has been widely exploited for many computer vision applications, including image classification, image captioning, pose estimation, etc. SAGAN [36] utilizes the self-attention module presented in [37], to capture long-range dependencies. With this attention mechanism, the long-range dependencies are embedded in the feature maps. It is complementary to the convolutions, whose advantage lies in modeling local dependencies. Different from the self-attention module used in SAGAN [36], DSiamM [12] learns a spatial weight maps to fuse multi-level deep features. More specifically, when the target is near the center of the search region, the shallower layer features get more weight at the center region while the background region gets less weight. The weight of the deeper layer is the opposite as illustrated in Figure 2.10. Owing to the fact that features from the deeper layer help to remove the background interference while features from the shallower layer can get precise localization of the target, the complementary role of response maps from different layers is reflected in the attention maps. It is helpful to obtain a better tracking performance. Besides, another way to take advantage of visual attention is exploited in [38]. They regard object tracking as a two-stage problem, drawing some samples in the search image first and classifying these samples as target or background. The attention maps they used are the partial derivative output of the first layer, which indicates the importance of each pixel in the input sample to affect the classification accuracy. Through a reciprocative learning scheme, the attention maps cover the whole target region gradually, and the tracking performance is improved even if target objects undergo large movements.

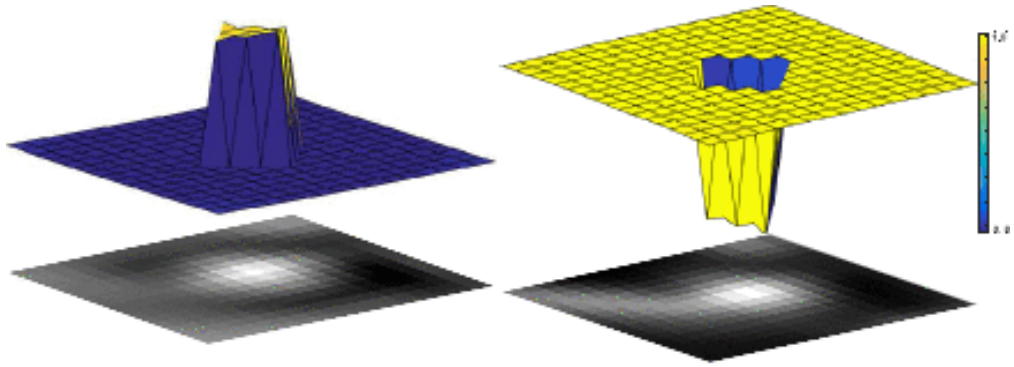


Figure 2.10 The spatial weight maps learned in DSiamM [12], the left map is learned for the shallower layer of AlexNet, the right map is learned for the deeper layer of AlexNet. (This image is adapted [reprinted] from [12] © 2017 IEEE)

## 2.5 Evaluation Metrics

Wu et al. [1] proposed the OTB benchmark, which has been widely used to evaluate state-of-the-art trackers. It consists of three datasets, OTB-2013, OTB-50 and OTB-100. The OTB-2013 dataset contains sequences with 51 target objects, the OTB-50 and the OTB-100 include sequences with 50 target objects and 100 target objects, respectively. All sequences are categorized according to 11 attributes as illustrated in Table 1.1. The benchmark considers two evaluation metrics, the center location error and the overlap score.

The center location error is known as the Precision plot, which computes the average squared error between the center locations of the ground-truth labels and the tracked targets of all the frames. Since the average error is prone to be affected by bias results, it does not measure the tracking performance with much accuracy. Thus, the percentage of frames in which the precision is within a given threshold is reported. The precision at the threshold of 20 pixels is typically used to rank trackers.

The overlap score is known as the Success plot, which computes intersection over union (IoU) between the ground-truth labels and the tracked targets. Given a score threshold, the overlap scores demonstrate whether a tracker successfully tracks the target object in one frame. As the threshold varies between 0 and 1, the corresponding success rates are drawn as the Success plot. The area under curve (AUC) of each success plot is usually used to rank trackers, which corresponds to the average success rates for all sampled overlap thresholds.

## 2.6 Discussion

In this chapter, we introduced the process of VOT and the background knowledge. In addition, we also discussed some the state-of-the-art approaches from two perspective, correlation filter-based trackers and deep learning-based trackers. Finally, we presented the standard evaluation metrics used for VOT.

Correlation with an example template is a common way to detect target object in VOT. Besides, with the help of kernel trick, the tracking speed of most KCF-based trackers [5, 19, 20] has been increased by an order of magnitude. The correlation filter works well if the appearance of the tracking target does not change a lot. Whereas, in most visual object tracking scenario, the target object undergoes significant appearance changes. To improve the robustness of tracking, many works [6–8] train convolutional neural networks to represent features. In the meantime, some works [9, 21] combine CNNs with correlation filters, alleviating the depth and the number of parameters of CNNs. However, their performances are still limited when target objects undergo large movements. Since there are lots of uncertainties in the tracking objects, only a single pretrained networks can not generate discriminative enough feature representations. To optimize the representation ability of CNNs in VOT, we explore strategies including multiple branches, online selection, and hierarchical feature fusing, in our work. By using multiple branches, diverse representations can be generated. For different tracking situations, we can choose the optimal representation via our online selection mechanism against the target appearance changes. In addition, combining hierarchical convolutional features to encode the target can enhance the representation power comparing with features from the only last convolutional layer.

# CHAPTER 3 OVERVIEW

In this thesis, we propose approaches to address challenges in visual object tracking. Specifically speaking, in visual object tracking scenarios, target object appearance can be affected by object motion, changes in viewpoint, lighting variation, and other disturbing factors. In our work, we employ a more robust and efficient feature representation against those disturbing factors. The proposed methods are described in Chapter 4 and Chapter 5.

## 3.1 Multi-Branch Siamese Networks with Online Selection

In our first article, we present MBST, a multi-branch Siamese tracker. Considering that the feature representation ability is not robust enough especially when facing those disturbing factors mentioned above, our goal is to explore strategies to enrich the feature representation of neural networks-based trackers. Hence, we proposed two strategies to improve the discriminative ability of pretrained models. The first one is training the network in different contexts, corresponding to different context-dependent branches. Since each branch is fine-tuned in a different context, they can have different performance in different tracking scenarios. The second strategy is to add networks designed and trained for different tasks. In our approach, we utilize the pretrained AlexNet [23], which is designed for image classification task and trained in ImageNet [24] dataset.

After we get multiple branches, different branches have different performance in different tracking scenarios. Combining these different branches can diversify the feature representation of our approach. Thus, we need to find a way to ensemble these branches together and maximize the tracking performance of the whole network. Given more discriminative response maps usually generate more accurate tracking results, we proposed an online branch selection mechanism to select branches when tracking. The online branch selection mechanism selects the branch to be used in tracking according to the response maps which are calculated by the feature representations generated in these branches. Besides, to deal with the problem of target scale variations, we input the target object over three scales and update the size of the bounding box by the scale obtaining the maximum response value.

With our multiple branches architecture and online branch selection mechanism, our tracker obtains improved performance compared to standard Siamese network trackers on object tracking benchmarks [1, 13].

### 3.2 Multiple Features of Siamese Networks

In the second paper, we improve the discriminative power of feature representations by utilizing features from different convolutional layers and two CNN models and using a feature recalibration mechanism. A Multiple Features-siamese Tracker (MFST) is proposed in this paper. It is well known that deep features from the last convolutional layer embed semantic information of the target, which is invariant to appearance changes. However, since the receptive field of the convolutional features from the deeper layer is large, it cannot get a precise spatial localization which is very important in object tracking. On the contrary, features from shallower layers contain more details and can get more precise localization but they are prone to fail when appearance changes. Besides, as we observed from multi-branch tracking methods, the combination of multiple CNN models can diversify the feature representation, to ensure a more efficient and robust tracking. To make better use of features extracted from the CNN models, we combine multiple features from different convolutional layers and different CNN models. To combine these features appropriately, we exploit three types of combination schemes, hard weight, soft mean and soft weight. The optimal strategy is used in our tracker to obtain the best feature representation of the target object.

In addition, we employ a feature recalibration mechanism in our approach, the Squeeze-and-Excitation block. It extracts the channel descriptor first and then applies a two layers MLP to learn the channel-wise dependencies. Since different channels of the features play different roles in tracking, by using the feature recalibration mechanism, the discriminative channels of the features are enhanced while the others are weakened.

With the combination of hierarchical features from different models and the feature recalibration mechanism, a much-improved feature representation is obtained with our method. The MFST achieves comparable performance among some recent state-of-the-art tracker on the OTB benchmarks [1, 13].

### 3.3 Experimentation

To validate the robustness of feature representation proposed in both of our trackers, we evaluated our methods on the visual object tracking benchmarks OTB2013 [13], OTB50 [13] and OTB100 [1]. After training our models offline, the learned models are used for tracking. The evaluation experiments were executed multiple times, which ensures the reliability of results. We also did an ablation study to assess the contribution of each module in our method. In addition, our method was evaluated in different tracking scenarios with different challenges. More details are presented in Chapter 4 and Chapter 5.

# CHAPTER 4 ARTICLE 1: MULTI-BRANCH SIAMESE NETWORKS WITH ONLINE SELECTION FOR OBJECT TRACKING

**Authors**

Zhenxi Li, Guillaume-Alexandre Bilodeau,
*LITIV lab, Polytechnique Montreal*
Wassim Bouachir,
*TELUQ University*

E-mail: {zhenxi.li, guillaume-alexandre.bilodeau}@polymtl.ca,wassim.bouachir@teluq.ca

**Published in** International Symposium on Visual Computing, ISVC, November, 2018

**Reference as** Li, Z., Bilodeau, G.-A., Bouachir, W., Multi-Branch Siamese Networks with Online Selection for Object Tracking. International Symposium on Visual Computing (ISVC), Las Vegas, NV, USA, November 19-21, 2018, pp. 309-319

## 4.1 Abstract

In this paper, we propose a robust object tracking algorithm based on a branch selection mechanism to choose the most efficient object representations from multi-branch siamese networks. While most deep learning trackers use a single CNN for target representation, the proposed Multi-Branch Siamese Tracker (MBST) employs multiple branches of CNNs pre-trained for different tasks, and used for various target representations in our tracking method. With our branch selection mechanism, the appropriate CNN branch is selected depending on the target characteristics in an online manner. By using the most adequate target representation with respect to the tracked object, our method achieves real-time tracking, while obtaining improved performance compared to standard Siamese network trackers on object tracking benchmarks.

***Keywords:*** Object tracking, Siamese networks, Online branch selection.

## 4.2 Introduction

Model-free visual object tracking is one of the most fundamental problems in computer vision. Given the object of interest marked in the first video frame, the objective is to localize the target in subsequent frames, despite object motion, changes in viewpoint, lighting variation, among other disturbing factors. One of the most challenging difficulties with model-free tracking is the lack of prior knowledge on the target object appearance. Since any arbitrary object may be tracked, it is impossible to train a fully specialized tracker.

Recently, convolutional neural networks (CNNs) have demonstrated strong power in learning feature representations. To fully exploit the representation power of CNNs in visual tracking, it is desirable to train them on large datasets specialized for visual tracking, and covering a wide range of variations in the combination of target and background. However, it is truly challenging to learn a unified representation based on videos that have completely different characteristics. Some trackers [7] train regression networks for tracking in an entirely offline manner. Other works [8, 9, 34] propose to train deep CNNs to address the general similarity learning problem in an offline phase and evaluate the similarity online during tracking. However, since these works have no online adaptation, the representations they learned offline are general but not always discriminative.

Rather than applying a single fixed network for feature extraction, we propose to use multiple network branches with an online branch selection mechanism. It is well known that different networks designed and trained for different tasks have diverse feature representations. With the online branch selection mechanism, our tracker dynamically selects the most efficient and robust branch for target representation, even if the target appearance changes. Our goal is to improve the generalization capability with multiple networks.

The main contributions of our work are summarized as follows. First, we propose a multi-branch framework based on a siamese network for object tracking. The proposed architecture is designed to extract appearance representation robust against target variations and changing contrast with background scene elements. Second, to make the full use of the different branches, we propose an effective and generic branch selection mechanism to dynamically select branches according to their discriminative power. Third, on the basis of multiple branches and branch selection mechanism, we present a novel deep learning tracker achieving real-time and improved tracking performance. Our extensive experiments compare the proposed Multi-Branch Siamese Tracker (MBST) with state-of-the-art trackers on OTB benchmarks [1, 13].

### 4.3 Related Work

**Siamese Network Based Trackers.** Object tracking can be addressed using similarity learning. By learning a deep embedding function, we can evaluate the similarity between an exemplar image patch and a candidate patch in a search region. These procedures allow to track the target to the location that obtains the highest similarity score. Inspired by this idea, the pioneering work of SiamFC [8] proposed a fully-convolutional Siamese Network in which the similarity learning with deep CNNs is addressed using a Siamese architecture. Since this approach does not need online training, it can easily achieve real-time tracking. Due to the robustness and real-time performance of the SiamFC [8] approach, several subsequent works proceeded along this direction to address the tracking problem. In this context, EAST [39] employs an early-stopping agent to speed up tracking where easy frames are processed with cheap features, while challenging frames are processed with deep features. CFNet [9] incorporates a Correlation Filter into a shallow siamese network, which can speed up tracking without accuracy drop comparing to a deep Siamese network. TRACA [14] applies context-aware feature compression before tracking to achieve high tracking performance. SA-Siam [34] utilizes the combination of semantic features and appearance features to improve generalization capability. In our work, we use the Siamese Network as embedding function to extract feature representations. All branches use the Siamese architecture to apply identical transformation on target patch and search region.

**Multi-Branch Tracking Frameworks.** The diversity of target representation from a single fixed network is limited. The learned features may not be discriminative in all tracking situations. There are many works using diverse features with context-aware or domain-aware scheme.

TRACA [14] is a multi-branch tracker, which utilizes multiple expert auto-encoders to robustly compress raw deep convolutional features. Since each of expert auto-encoders is trained according to a different context, it performs context-dependent compression. MDNet [33] is composed of shared layers and multiple branches of domain-specific layers. BranchOut [40] employs a CNN for target representation, with a common convolutional layers and multiple branches of fully connected layers. It allows different number of layers in each branch to maintain variable abstraction levels of target appearances.

A common insight of these multi-branch trackers is the possibility to make a robust tracker by utilizing different feature representations. Our method shares some insights and design principles with other multi-branch trackers. Our network architecture is composed of multiple branches separately trained offline and focusing on different types of CNN features. In

addition, we use an AlexNet [23] branch in our framework that is designed and pretrained for image classification. In our multi-branch frameworks, the combination of branches trained in different scenarios ensures a better use of diverse feature representations.

**Online Branch Selection.** Different models produce various feature maps on different tracked targets in different scales, rotations, illumination and other factors. Using all features available for a single object tracking is neither efficient nor effective. BranchOut [40] selects a subset of branches randomly for model update to diversify learned target appearance models. MDNet [33] learns domain-independent representations from pretraining, and identifies branches through online learning.

In our online branch selection mechanism, we analyse the feature representation of each branch to select the most robust branch at every $T$ frames. This allows us to use diverse feature representations and to handle various challenges in the object tracking problem more efficiently.

## 4.4 Multi-Branch Siamese Tracker

We propose a multi-branch siamese network for tracking. Given that different neural network models produce diverse feature representations, we use many of them as branches in our tracker to produce diverse feature representations and select the most robust branch with our online branch selection mechanism.

### 4.4.1 Network Architecture

Using multiple target representations is shown to be beneficial for object tracking [34, 41], as different CNNs can provide various feature representations. In our work, we ensemble $N_e$ siamese networks including $N_s$ context-dependent branches and one AlexNet branch as $N_e = N_s + 1$. The context-dependent branches have the same structure as SiamFC [8] and the AlexNet branch has the same structure as AlexNet [23]. Each branch of the tracker is a siamese network applying identical transformation $\varphi_i$ to both inputs and combining their representation by a cross-correlation layer. The architecture of the proposed tracker is illustrated in Fig. 4.1.

The input consists of a target patch cropped from the first video frame and another patch containing the search region in the current frame. The target patch $z$ has a size of $W_z \times H_z \times 3$, corresponding to the width, height and color channels of the image patch. The search region $X$ has a size of $W_X \times H_X \times 3$ ($W_z < W_X$ and $H_z < H_X$), representing also the width, height and color channels of the search region. $X$ can be considered as a collection of candidate

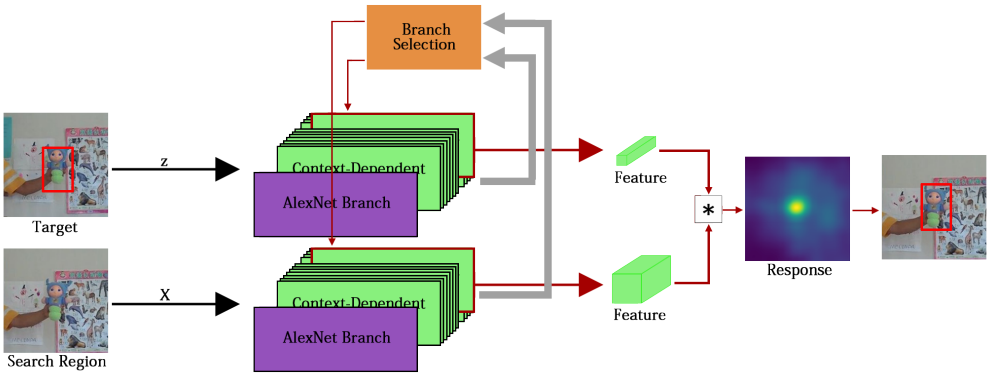



Figure 4.1 The architecture of our MBST tracker. Context-dependent branches are indicated by green blocks and AlexNet branch is indicated by purple blocks.

patches $x$ in the search region with the same dimension as $z$.

From what we observed, there are two strategies to improve the discriminative ability of the tracking networks. The first one is training the network in different contexts, while the second one is to use multiple networks designed and trained for different tasks. In our approach, we utilize context-dependent branches pretrained in different contexts in addition to another branch pretrained for image classification task to improve our tracking performance. We note that more branches could be added with other pre-trained networks at the cost of slower performances.

**Context-dependent branches:** We use $N_c$ context-dependent branches and one general branch as $N_s = N_c + 1$. All these branches have the same architecture as the SiamFC network [8]. Context-dependent branches are trained in three steps. Firstly, we train the basic siamese network on the ILSVRC-2015 [24] video dataset (henceforth ImageNet), including 4,000 video sequences and around 1.3 million frames containing about 2 million tracked objects. We keep the basic siamese network as the general branch. Then, we perform contextual clustering on the low level feature map from the ImageNet Video dataset to find $N_c$ ($N_c = 10$) context-dependent clusters. Finally, we use the $N_c$ clusters to train $N_c$ context-dependent branches initialized by the basic siamese network. These branches take $(z, X)$ as input and extract their feature maps. Then, using a cross correlation layer we combine their feature maps to get a response map. The response map of context-dependent branches is calculated as:

$$h_{s_i}(z, X) = corr(f_{s_i}(z), f_{s_i}(X)), \tag{4.1}$$

where $s_i$ indicates the contextual index including the general branch ($i = 0$), $f(\cdot)$ denotes

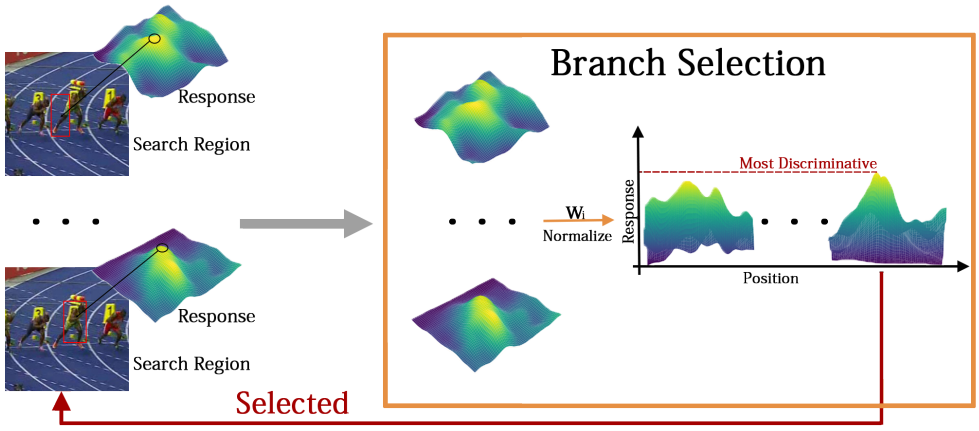



Figure 4.2 Online branch Selection mechanism and response map example.

features generated by the network.

**The AlexNet branch:** We use AlexNet [23] pretrained on the image classification task as a branch with a network trained for a different task. Small modifications are made on the stride to ensure that the output response map has the same dimension as other branches. Since AlexNet is trained for image classification and the deeper layers encode more semantic information of targets, target representations from this branch are more robust to significant appearance variations. The network output corresponds to $(z, X)$ as input, while the generated features are denoted as $f_a(\cdot)$. The response map is expressed as:

$$h_a(z, X) = corr(f_a(z), f_a(X)). \tag{4.2}$$

In our implementation, MBST is composed of context-dependent branches and AlexNet branch. The output of each branch is a response map indicating the similarity between target $z$ and candidate patch $x$ within the search region $X$. The branch selection mechanism compares the maps from each branch to select the most discriminative one. The corresponding branch is then used for $T - 1$ frames.

### 4.4.2 Online Branch Selection Mechanism

Different branches trained in different scenarios can be used to diversify the target representation. To ensure the optimal exploitation of the diverse representations from our branches, we designed a branch selection mechanism to monitor the tracking output and automatically select the most discriminative branch as illustrated in Fig. 4.2.

Given the input image pair, each branch applies identical transformation to both inputs and

calculates the response map $h$ using a cross-correlation layer. Since the ranges of feature values from different branches are different, we apply response weights $w_i$ on response map of each branches to normalize their range difference. The discriminative power is then measured based on the weighted response maps from all branches. The heuristic approach we used to measure the discriminative power of branches is formulated as:

$$R(w_i h_{B_i}) = w_i(P(h_{B_i}) - M(h_{B_i})), \tag{4.3}$$

where $h_{B_i}$ is the response map for each branch $B_i$, $P_{B_i}$ is the peak value of the response map $h_{B_i}$, and $M_{h_{B_i}}$ is the minimum value of the response map $h_{B_i}$.

The objective function of our branch selection mechanism can be written as:

$$B^* =_{B_i} R(w_i h_{B_i}), \tag{4.4}$$

where $B^*$ is the selected branch to transform inputs.

## 4.5 Experiments

The first aim of our experiments is to investigate the effect of incorporating multiple feature representations with an online branch selection mechanism. For this purpose, we performed ablation analysis on our framework. We then compare our method with state-of-the-art trackers. The experimental results demonstrate that our method achieves improved performance with respect to the basic SiamFC tracker [8].

### 4.5.1 Implementation Details

**Network structure:** The context-dependent branches have exactly the same structure as the SiamFC network [8]. For the AlexNet branch, we use AlexNet [23] pretrained on ImageNet dataset [24] with a small modification to ensure that the output response map has the same dimension as other branches, which is $17 \times 17$. Other branches could also be used based on other network architectures.

**Data Dimensions:** In our experiment, the target image patch $z$ has a dimension of $127 \times 127 \times 3$, and the search region $X$ has a dimension of $255 \times 255 \times 3$. But since all branches are fully convolution layers, they can also be adapted to any other dimension easily. The embedding output for $z$ and $X$ has a dimension of $6 \times 6 \times 256$ and $22 \times 22 \times 256$ respectively.

**Training:** We use the ImageNet dataset [24] for training and only consider color images.

For simplicity, we randomly pick a pair of images, we crop $z$ in the center and $X$ in the center of another image. Images are scaled such that the bounding box, plus an added margin for context, has a fixed area. The basic siamese branch is trained for 50 epochs with an initial learning rate of 0.01. The learning rate decays after every epoch with a decay factor $\delta$ of 0.869. The context-dependent branches are fine-tuned based on the parameters of the general branch with a learning rate 0.00001 for 10 epochs. For the AlexNet branch, we directly use AlexNet [23] pretrained on ImageNet dataset [24].

Our experiments are performed on a PC with a Intel i7-3770 3.40 GHz CPU and a Nvidia Titan X GPU. We evaluated our results using the Python implementation of the OTB toolkit. The average testing speed of MBST is 17 fps.

**Hyperparameters:** The weights $w_i$ for context-dependent branches have the same value of 1.0. For AlexNet branch, we perform a grid search from 8.0 to 12.0 with step 0.5. Evaluation suggests that the best performance is achieved when $w_i$ is 10.5. This value is thus used for all the test sequences. In order to handle scale variations, we rescale the inputs into three different resolutions.

### 4.5.2 Dataset and Evaluation Metrics

**OTB:** We evaluate the proposed tracker on the OTB benchmarks [1,13] with eleven interference attributes for the video sequences. The OTB benchmark uses the precision and success rate for quantitative analysis. For the precision plot, we calculate the average Euclidean distance between the center locations of the tracked targets and the manually labeled ground truth. Then the average center location error over all the frames of one sequences is used to summarize the overall performance. As the representative precision score for each tracker, we use the score for the threshold of 20 pixels. For the success plot, we compute the IoU (in-

Table 4.1 Ablation study of MBST on OTB benchmarks. Various combinations of general siamese branch, context-dependent branches and AlexNet branch are evaluated.

| | | | OTB-2013 | | OTB-50 | | OTB-100 | | |
|---|---|---|---|---|---|---|---|---|---|
| General | Context | AlexNet | AUC | Prec. | AUC | Prec. | AUC | Prec. | FPS |
| ✓ | | | 0.600 | 0.791 | 0.519 | 0.698 | 0.585 | 0.766 | **65.0** |
| | ✓ | | 0.601 | 0.798 | 0.523 | 0.707 | 0.584 | 0.768 | 18.6 |
| | | ✓ | 0.581 | 0.761 | 0.501 | 0.678 | 0.560 | 0.741 | 63.6 |
| ✓ | ✓ | | 0.594 | 0.784 | 0.535 | 0.721 | 0.587 | 0.770 | 16.9 |
| ✓ | | ✓ | 0.605 | 0.796 | 0.536 | 0.718 | 0.599 | 0.783 | 42.9 |
| | ✓ | ✓ | 0.616 | 0.811 | 0.570 | 0.767 | 0.614 | 0.806 | 16.9 |
| ✓ | ✓ | ✓ | **0.620** | **0.816** | **0.573** | **0.773** | **0.617** | **0.811** | 16.9 |

tersection over union) between the tracked and ground truth bounding boxes. A success plot is obtained by evaluating the success rate at different IoU thresholds. The area-under-curve (AUC) of the success plot is reported.

### 4.5.3 Ablation Analysis

To verify the contribution of each branch and the online branch selection mechanism of our algorithm, we implemented several variations of our approach and evaluated them on the OTB benchmarks.

**Multiple branches improve the tracking result.** We compared our full branches algorithm with various combination of branches as illustrated in Table 4.1. We evaluate the performances of the original branch, context-dependent branches and AlexNet branch alone. Note that branch selection is applied only when we evaluate the context-dependent branches, since many branches are available. For the other experiments in Table 4.1, we combine these branches with online branch selection for testing. Results clearly demonstrate that the proposed multiple branches architecture allows a better use of diverse feature representations. The best FPS is achieved by the general siamese branch, which is expected since it needs less computations with only one branch.

**Online branch selection for every frame is not necessary.** As shown in Fig. 4.3, we conduct experiments on the branch selection interval $T$ by changing the value: $T = 1, 3, 5, 7, 10, 13$. When the value of branch selection interval is less than 7 frames, the tracking performance is reduced. This can be explained by the fact that a frequent execution of the selection mechanism increases the possibility of selecting an inappropriate branch. When the value of branch selection interval is more than 7 frames, the tracking performance is also decreased because we keep for a too long period a branch that is not discriminative anymore.

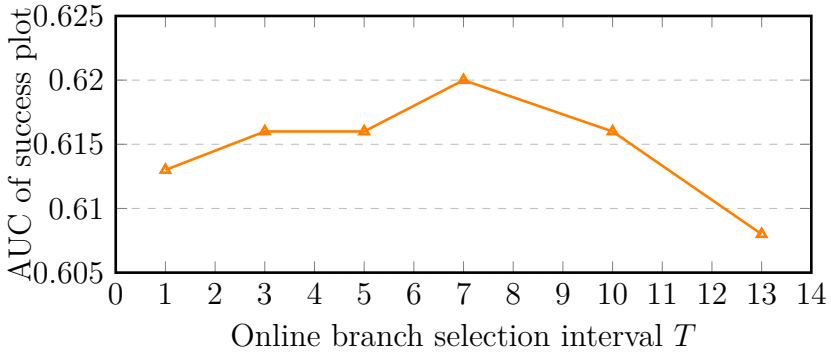



Figure 4.3 Curve for the branch selection interval $T$ on OTB2013 benchmark [13].

In our experiments, the optimal value of branch selection interval $T$ was 7 frames.

### 4.5.4 Comparison with State-of-the Art Trackers

We compare MBST with CFNet [9], SiamFC [8], Staple [42], LCT [43], Struck [44], MEEM [45], SCM [46], LMCF [47], MUSTER [48], TLD [49] on OTB benchmarks. The precision plots and success plots of one path evaluation (OPE) are shown in Fig. 4.4. Based on precision and success plots, the overall comparison suggests that the proposed MBST achieved the best performance among these state-of-the-art trackers on OTB benchmarks. Notably, it outperforms SiamFC [8] as well as its variation CFNet [9] on all datasets. This demonstrates that diverse feature representations are important to improve tracking, as feature maps from various CNNs can be quite different. Fig. 4.5 demonstrates that our tracker effectively handles all kinds of challenging situations that often require high-level semantic understanding. For example, our tracker significantly outperforms SiamFC in the case of deformation, occlusion and out-of-plane rotations because the contrast between the object and the background changes and switching to another feature map may give a better discriminativity. Therefore, our approach is beneficial each time the appearance of the object changes significantly during its tracking.

## 4.6 Conclusion

In this paper, we propose a Multi-Branch Siamese Network with Online Selection. We ensemble multiple siamese networks to diversify target feature representations. Using our online branch selection mechanism, the most discriminative branch is selected against target appearance variations. Our tracker benefits from the diverse target representation, and can handle all kinds of challenging situations in visual object tracking. Our experiment results show improved performances compared to standard Siamese network trackers, while outperform several recent state-of-the-art trackers.

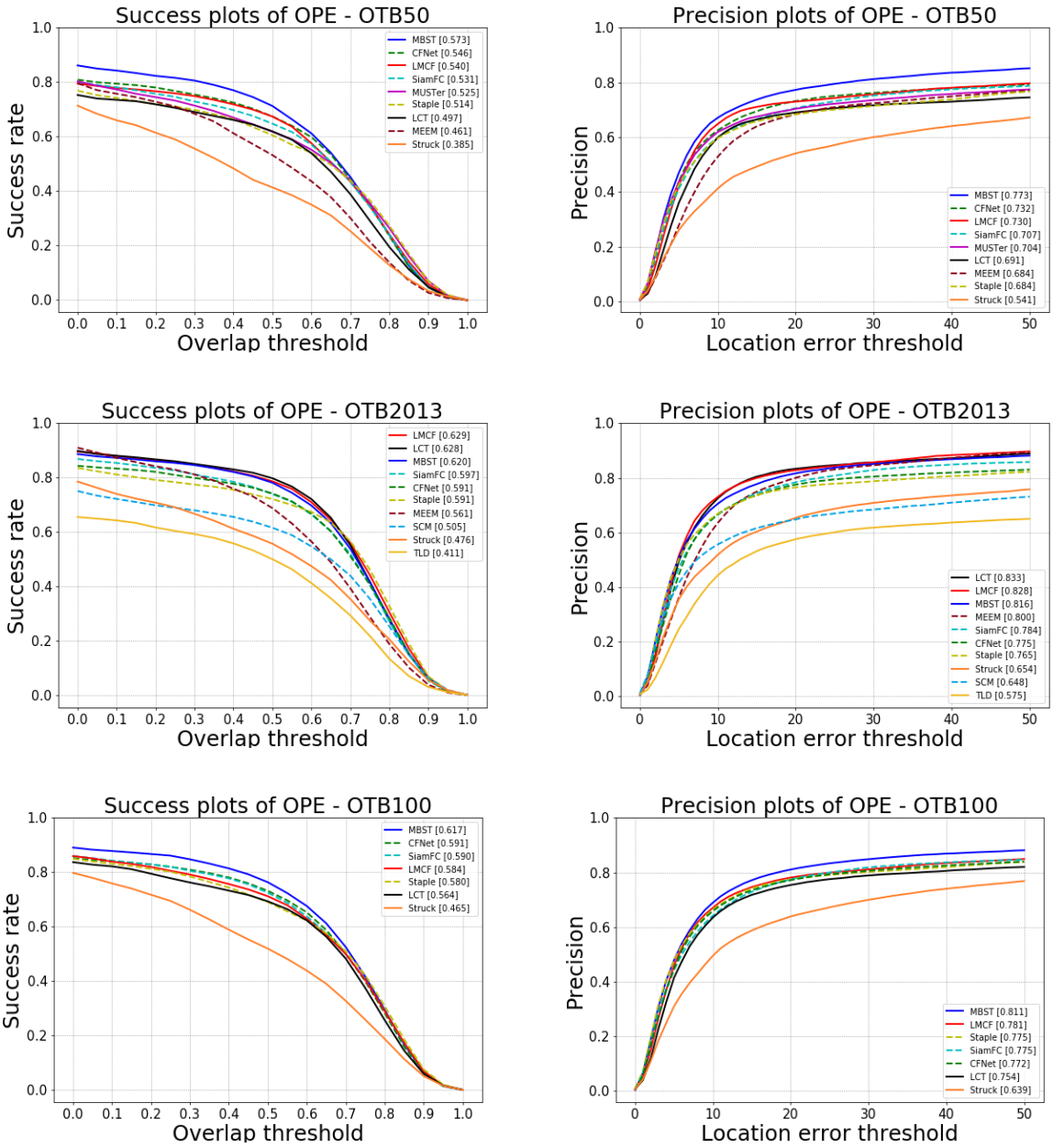



Figure 4.4 The success plots and precision plots on OTB benchmarks. Curves and numbers are generated with Python implemented OTB toolkit.

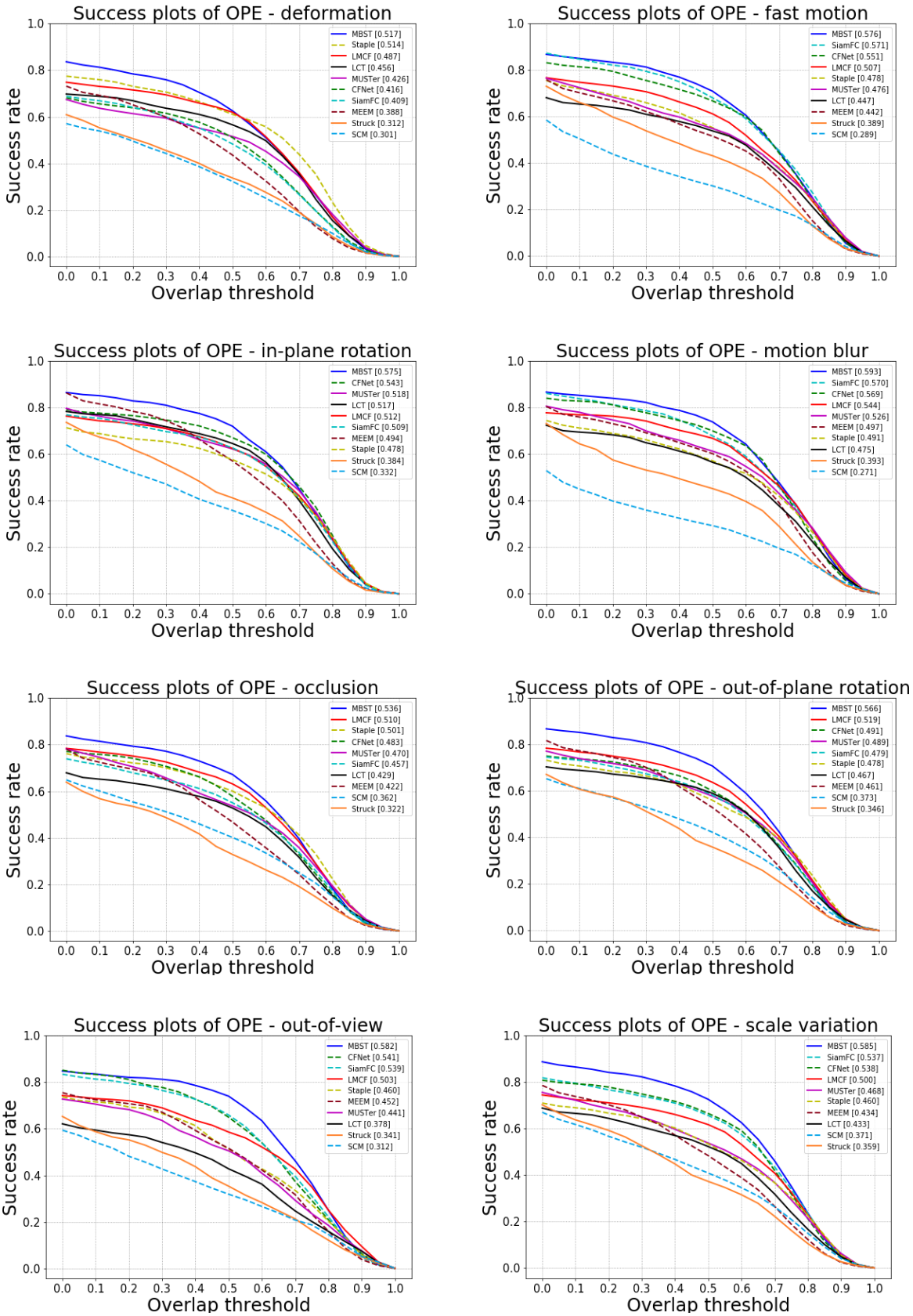



Figure 4.5 The Success plot on OTB50 for eight challenge attributes: deformation, fast motion, in-plane rotation, motion blur, occlusion, out-of-plane rotation, out-of-view, scale variation.

# CHAPTER 5 ARTICLE 2: MULTIPLE CONVOLUTIONAL FEATURES IN SIAMESE NETWORKS FOR OBJECT TRACKING

**Authors**

Zhenxi Li, Guillaume-Alexandre Bilodeau,
*LITIV lab, Polytechnique Montreal*
Wassim Bouachir,
*TELUQ University*

E-mail: {zhenxi.li, guillaume-alexandre.bilodeau}@polymtl.ca,wassim.bouachir@teluq.ca

**Submitted to** Computer Vision and Image Understanding, CVIU, February, 2019

## 5.1 Abstract

Recently, Siamese trackers demonstrated high performance in object tracking due to their balanced accuracy-speed. Unlike classification-based CNNs, deep similarity networks are specifically designed to address the image similarity problem, and thus are inherently more appropriate for the tracking task. However, Siamese trackers mainly use the last convolutional layers for similarity analysis and target search, which restricts their performance. In this paper, we argue that using a single convolutional layer as feature representation is not the optimal choice within the deep similarity framework. We present a Multiple Features-Siamese Tracker (MFST), a novel tracking algorithm exploiting several hierarchical feature maps for robust deep similarity tracking. Since Convolutional layers provide several abstraction levels in characterizing an object, fusing hierarchical features allows to ensure a richer and more efficient representation. Moreover, we handle the target appearance variation by calibrating deep features extracted from two different CNN models. Based on this advanced feature representation, our method achieves high tracking accuracy, while outperforming standard Siamese trackers on object tracking benchmarks.

***Keywords:*** object tracking, siamese networks, feature combination, hierarchical convolutional features, feature representation, deep learning

## 5.2 Introduction

Visual object tracking (VOT) is one of the most fundamental tasks in computer vision. Given a target object in the first frame, the objective of visual object tracking is to determine the object state in the following frames. With the rapid development of computer vision, visual object tracking has been employed in many applications, such as autonomous driving, visual analysis and video surveillance. For example, with the help of visual object tracking, autonomous driving systems can predict obstacle movements and decide where to go.

During the last years, deep learning trackers achieved stimulating, by bringing new ideas to object tracking. This paradigm has become successful mainly due the use of convolutional neural network (CNN)-based features for appearance modeling and their discriminative ability to represent target objects. While most tracking methods used classification-based CNN models, that are built following the principals of visual classification tasks, another approach [8] formulated the tracking task as a deep similarity learning problem, where a Siamese network in trained to locate the target within a search image. This method uses feature representations extracted by CNNs and performs correlation operation with a sliding window to calculate a similarity map for target location. Rather than detecting by correlation, other deep similarity trackers [7, 22, 35] generate the bounding box for the target object by regression networks. For example, GOTURN [7] predicts the bounding box of the target object with a simple CNN model. The trackers [35] and [22] generate a number of proposals for the target after extracting feature representations. Classification and regression procedures are then applied to produce the final tracking.

By formulating object tracking as a deep similarity problem, Siamese trackers achieved significant progress in terms of both speed and accuracy. However, less efforts have been devoted to advance the feature representation power of these models. Siamese trackers typically use only features from the last convolutional layers for similarity analysis and target state prediction. We argue that this is not the optimal choice and demonstrate that features from earlier layers are also beneficial for more accurate deep similarity tracking. Indeed, the combination of different convolutional layers was shown to be efficient for robust tracking [11, 21]. As we go deeper in a CNN, the receptive field becomes wider, therefore, features from different layers contain different levels of information. In this way, the last convolutional layers retain general characteristics represented in summarized fashion, while the first convolutional layers provide low-level features. These latter are extremely valuable for precise localization of the target, as they are more object-specific and capture spatial details. Furthermore, instead of using features from a single CNN model, we propose to exploit different models within the deep similarity framework. Diversifying feature representations significantly improves

tracking performance. Such strategy is shown to ensure a better robustness against target appearance variation, one of the most challenging tracking difficulties [50].

Based on these principals, we propose a Multiple Features-Siamese Tracker (MFST). Our tracker utilizes diverse features from several convolutional layers and two models with proper feature fusing strategies to achieve an improved tracking performance. Our contributions can be summarized as follows. Firstly, we explore feature fusing strategies with a feature recalibration module to make a better use of feature representations. Secondly, we exploit feature representations from several hierarchical convolutional layers as well as different models for object tracking. Thirdly, we present the MFST tracking algorithm, that achieves strong performance with respect to recent state-of-the-art trackers on popular OTB benchmarks.

The paper is organized as follows. We present related work in Section 5.3, the proposed MSFT tracker in Section 5.4, and the experimental results in Section 5.5 respectively. Finally, Section 5.6 concludes the paper.

### 5.3 Related Work

**Siamese Trackers** VOT can be formulated as a similarity learning problem. Once the deep similarity network is trained during an offline phase to learn a general similarity function, the model is applied for online tracking, by analyzing the similarity between the two network inputs: the target template and the current frame. The pioneering work SiamFC [8] applied two identical branches made up of fully convolutional neural networks to extract the feature representations, on which cross correlation is computed to generate the tracking result. SiamFC outperformed most of the best trackers at that time, while achieving real-time speed. Rather than performing correlation on deep features directly, CFNet [9] trains a correlation filter based on the extracted features of object to speed up tracking without drop in accuracy. MBST [50] improved the tracking performance by using multiple siamese networks as branches to enhance the diversity of the feature representation. SA-Siam [34] encodes the target by a semantic branch and an appearance branch to improve the robustness of tracking. However, since these siamese trackers only take the output of the last convolutional layers, more detailed target specific information from earlier layers is not used. In our work, we adopt a Siamese architecture to extract deep features for the target and search region, but combine features from different layers of networks for tracking. With multiple levels of abstraction embedded, the target representation becomes more efficient.

**Hierarchical Convolutional Features in Tracking** CNNs have been commonly used in many computer vision tasks owing to their discriminative feature representations. Most

CNN-based trackers used only use the output of the last convolutional layer, containing semantic information represented in a summarized fashion. However, different convolutional layers embed different levels of visual abstraction. In fact, convolutional layers provide several detail levels in characterizing an object, and the combination of different convolutional levels is demonstrated to be efficient for robust tracking [21, 51]. In this context, the pioneering algorithm HCFT [21] tracks the target using correlation filters learned on each layer. With HCFT, the representation ability of hierarchical convolutional features is demonstrated to be better than features from a single layer. Subsequently, [11] presented a visualization of features extracted from different convolutional layers. In their work, they employed three convolutional layers as the target object representations, which are then convolved with the learned correlation filters to generate the response map, and a long-term memory filter to correct results. The use of hierarchical convolutional features allowed their trackers to be much more robust.

**Multi-Branch Tracking** One of the most challenging problem in object tracking is the varying appearance of the tracked objects. The appearance of the tracked objects may change a lot during tracking and for different scenarios. Thus, a single fixed networks cannot guarantee to generate discriminative feature representations in all tracking situations. To handle the problem of target appearance variations, TRACA [14] trained multiple auto-encoders, each for different appearance categories. These auto-encoders compress the feature representation for each category. The best expert auto-encoder is selected by a pretrained context-aware network. By selecting a specific auto-encoder for the tracked object, a more robust representation can be generated for the tracking. MDNet [33] applied a fixed CNN for feature extraction, but used multiple regression branches for objects belonging to different tracking scenarios. More recently, MBST [50] extracted the feature representation for the target object through multiple branches and selected the best branch according to their response maps. With multiple branches MBST can obtain diverse feature representations and select the most discriminative one under the prevailing circumstance. In their study, we can observe that the greater the number of branches, the more robust the tracker is. However, this is achieved at the cost of a higher computational time. In this work, we can get a diverse feature representation of a target at lower cost because some of the representations are extracted from the many layers of the CNNs. Therefore, we do not need a large number of siamese branches.

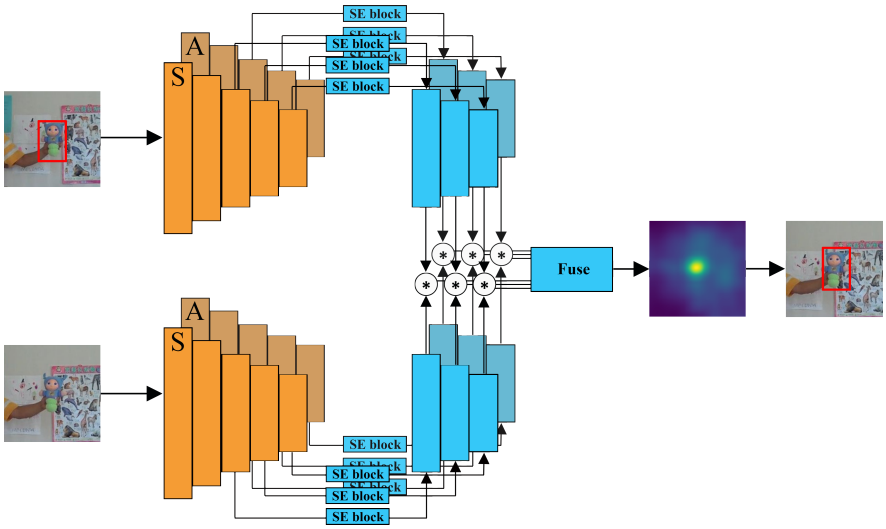



Figure 5.1 The architecture of our MFST tracker. Two CNN models are utilized as feature extractors and their features are calibrated by Squeeze-and-Excitation (SE) blocks. Then, correlations are applied over the features of the search region with the features of the exemplar patch and the output response maps are fused to calculate the new position of the target. Bright orange: SiamFC (S) and dark orange: AlexNet (A).

## 5.4 Multiple Features-Siamese Tracker

We propose a Multiple Features-Siamese Tracker for object tracking. For the design of our method, we considered that features from different convolutional layers contain different level of abstractions and that the different channels of the features play different roles in tracking. Therefore, we recalibrated deep features extracted from the CNN models and combined hierarchical features to make a more efficient representation. Besides, since models trained for different tasks can diversify the feature representation as well, we built our siamese architecture with two CNN models to achieve better performance.

### 5.4.1 Network Architecture

As many recent object tracking approaches [8,9,50] did, we formulate the tracking problem as a similarity learning problem and utilize the siamese architecture to address it. The network architecture of our tracker is shown in Figure 5.1. We use two different pretrained CNN models as feature extractors, SiamFC [8] and AlexNet [23], as indicated in Figure 5.1. The two models denoted as $S$ and $A$ respectively in the following explanations. Both of them are five layers fully convolutional neural networks.

The input of our method consists of an exemplar patch $z$ cropped according to the initial

bounding box or the result of last frame and search region $x$. The exemplar patch has a size of $W_z \times H_z \times 3$ and the search region has a size of $W_x \times H_x \times 3$ ($W_z < W_x$ and $H_z < H_x$), representing the width, height and the color channels of the image patches. Since our method formulates the tracking task as a similarity learning problem, let $x$ be considered as a collection of candidate patches.

With the two CNN models, we obtain the deep features $S_{l_i}$, $A_{l_i}$ ($l = c3, c4, c5$, $i = z, x$) from conv3, conv4, conv5 layers of each model. These are the preliminary deep feature representations of the inputs. Then, all the features are recalibrated through a Squeeze-and-Excitation blocks (SE-blocks) [52]. The recalibrated features are denoted as $S^*_{l_i}$, $A^*_{l_i}$, respectively for the two models. The details of SE-blocks is illustrated in Fig. 5.2. There blocks are trained to explore the importance of the different channels for tracking. They learn weights for the different channels to recalibrate features extracted from the preliminary models.

Once the feature representations are generated, we apply cross-correlation operations for each recalibrated feature map pairs to generate response maps. The cross-correlation operation can be implemented by a convolution layer using the feature of the exemplar as filter. Then we fuse these response maps to produce the final response map. The corresponding location of the maximum value in the response map is the new position of the target object.

### 5.4.2 Feature Extraction

**Hierarchical Convolutional Features** Deep learning has been widely used to solve computer vision problems. However, most works use the last convolutional layer to represent the image. It is well known that the last convolutional layer encode more semantic information which is invariant to significant appearance variations, compared to earlier layers. However, its resolution is too coarse (due to the large receptive field) for precise localization. On the contrary, features from earlier layers contain less semantic information, but they retain more spatial details and are more precise in localization. Thus, we propose to exploit multiple hierarchical levels of features to build a better representation of the target.

In our work, we use the convolutional layers of two pretrained CNN models as feature extractors: SiamFC [8] and AlexNet [23]. The two models are trained for object tracking and image classification tasks, respectively. We take features extracted from the 3rd, 4th, 5th layers as the preliminary target representations.

**Feature Recalibration** Considering that different channels of deep features play different roles in tracking, we apply SE-blocks [52] over the raw deep features extracted from the basic

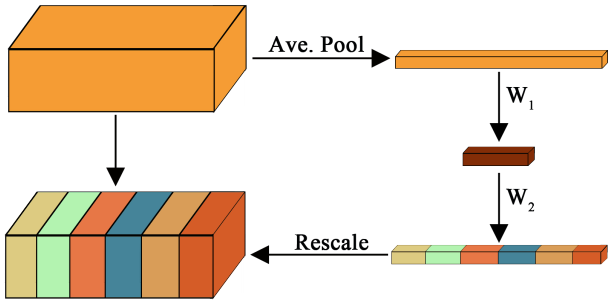



Figure 5.2 The illustration of SE-block. It consists of two step, squeeze step and excitation step. The squeeze step uses average pooling operation to generate the channel descriptor, the excitation step uses a two layers MLP to capture the channel-wise dependencies.

models. An illustration of SE-block is shown in Fig. 5.2. The SE-block consists of two steps: 1) squeeze and 2) excitation. The squeeze step corresponds to an average pooling operation. Given a 3D feature map, this operation generates the channel descriptor $\omega_{sq}$ as follows:

$$\omega_{sq} = \frac{1}{W \times H} \sum_{m=1}^{W} \sum_{n=1}^{H} v_c(m,n), (c = 1, ..., C) \tag{5.1}$$

where $W$, $H$, $C$ are the width, height and the number of channels of the deep feature, and $v_c(m,n)$ is the corresponding value in the feature map. The subsequent step is the Excitation through a two layers Multi-layer perceptron (MLP). Its goal is to capture the channel-wise dependencies that can be expressed as:

$$\omega_{ex} = \sigma(\mathbf{W_2}\delta(\mathbf{W_1}\omega_{sq})) \tag{5.2}$$

where $\sigma$ is a sigmoid activation, $\delta$ is a ReLU activation, $\mathbf{W_1} \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W_2} \in \mathbb{R}^{C \times \frac{C}{r}}$ are the weights for each layer, and $r$ is the channel reduction factor used to change the dimension. After the excitation operation, we obtain the channel weight $\omega_{ex}$. The weight is used to rescale the feature maps extracted by the basic models:

$$F_{l_i}^{*} = \omega_{ex} \cdot F_{l_i} \tag{5.3}$$

where $\cdot$ is a channel-wise multiplication and $F = (S, A)$. Note that $\omega_{ex}$ is learned for each layer in a basic model, but the corresponding layers for the CNNs of the exemplar patch and the search region use the same channel weight. We train the SE-blocks to obtain six $\omega_{ex}$ in total.

### 5.4.3 Response Maps Combination Mechanism

Once the feature representations from convolutional layers of each model are obtained, we apply cross-correlation operation, which is implemented by convolution, over the corresponding feature maps to generate the response map $r$ as:

$$r(z, x) = corr(F^*(z), F^*(x)) \tag{5.4}$$

where $F^*$ is the weighted feature maps generated by the CNN model and SE-block. The response maps are then combined. For a pair of image input, six response maps are generated, denoted as $r^S_{c3}$, $r^S_{c4}$, $r^S_{c5}$, $r^A_{c3}$, $r^A_{c4}$ and $r^A_{c5}$. Note that we do not need to rescale the response maps for combination, since they have the same size (see Section 5.5.1, Data Dimensions).

The response maps are combined hierarchically. After fusing $r^S$ and $r^A$ for the two CNN models, we combine the two response maps to get the final map. The combination is performed by considering three strategies: hard weight (HW), soft mean (SM) and soft weight (SM) [11], illustrated as follows:

$$\text{Hard weight: } r^* = \sum_{t=1}^{N} w_t r_t \tag{5.5}$$

$$\text{Soft mean: } r^* = \sum_{t=1}^{N} \frac{r_t}{max(r_t)} \tag{5.6}$$

$$\text{Soft weight: } r^* = \sum_{t=1}^{N} \frac{w_t r_t}{max(r_t)} \tag{5.7}$$

where $r^*$ is the combined response map, $N$ is the number of response maps to be combined together, and $w_t$ is the empirical weight for each response map. Finally, the corresponding location of the maximum value in the final response map is the new location of the target.

## 5.5 Experiments

The first objective of our experiments is to investigate the contribution of each module in order to find the best response map combination strategy corresponding to optimal representations. For this purpose, we perform an ablation analysis. Secondly, we compare our method with basic CNN model-based methods and recent state-of-the-art trackers. The experiment results show that our method significantly outperforms basic CNN-based methods, while obtaining competitive performance with respect to the recent state-of-the-art trackers.

### 5.5.1 Implementation Details

**Network Structure** We use SiamFC [8] and AlexNet [23] as deep feature extractors. The SiamFC network is a fully convolutional neural network, containing five convolutional layers. It has an AlexNet-like architecture, but it is trained on a video dataset for object tracking. The AlexNet network consists of five convolutional layers and three fully connected layers trained on an image classification dataset. We slightly modified the stride of AlexNet to obtain the same dimensions for the outputs of both CNN models. Since only deep features are needed to represent the target, we remove the fully connected layers of AlexNet and only keep the convolutional layers to extract features.

**Data Dimensions** The inputs of our method consist in the exemplar patch $z$ and the search region $x$. The size of $z$ is $127 \times 127$ and the size of $x$ is $255 \times 255$. The output feature maps of $z$ have sizes of $10 \times 10 \times 384$, $8 \times 8 \times 384$ and $6 \times 6 \times 256$ respectively. The output feature maps of $x$ have sizes of $26 \times 26 \times 384$, $24 \times 24 \times 384$ and $22 \times 22 \times 256$ respectively. Taking the features of $z$ as filters to perform a convolution on the features of $x$, the size of the output response maps are all the same, $17 \times 17$. The final response map is resized to the size of the input to locate the target. Since the two CNN models we used are all fully convolutional neural networks, the size of inputs can also be adapted to any other dimension.

**Training** The SiamFC model is trained on the ImageNet dataset ( [24]) and only color images are considered. The ImageNet dataset contains more that 4,000 sequences, about 1.3 million frames and 2 million tracked objects with ground truth bounding boxes. For the input, we take a pair of images and crop the exemplar patch $z$ in the center and the search region $x$ in another image. The SiamFC model is trained for 50 epochs with an initial learning rate of 0.01. The learning rate decays with a factor of 0.86 after each epoch. The AlexNet model is pretrained on the ImageNet dataset for the image classification task. We trained the SE-blocks for the two models separately in the same manner. For each model, the original parameters are fixed. We then apply SE-blocks on the output of each layer and take the recalibrated output of each layer as the output feature to generate the result for training. The SE-blocks are trained on the ImageNet dataset with 50 epochs with an initial learning rate of 0.01. The learning rate decays with a factor of 0.86 after each epoch.

We performed our experiments on a PC with an Intel i7-3770 3.40 GHz CPU and a Nvidia Titan X GPU. The benchmark results are calculated by the Python implementation of the OTB toolkit ( [1]). The average testing speed of our tracker is 39 fps.

**Hyperparameters** The channel reduction factor $r$ in SE-blocks is 4. The empirical weights $w_t$s for $r^S_{c3}$, $r^S_{c4}$, $r^S_{c5}$, $r^A_{c3}$, $r^A_{c4}$ and $r^A_{c5}$ are 0.1, 0.3, 0.7, 0.1, 0.6 and 0.3. The empirical weights $w_t$s for $r^S$ and $r^A$ are 0.3 and 0.7. To handle scale variations, we search the target object over three scales $1.025^{\{-1,0,1\}}$ during evaluation and testing.

### 5.5.2 Dataset and Evaluation Metrics

**OTB Benchmarks** We evaluate our method on the OTB benchmarks [1,13], which consist of three datasets, OTB50, OTB2013 and OTB100. They contain 50, 51, 100 video sequences with ground truth target labels for object tracking. The benchmarks proposed two evaluation metrics for quantitative analysis, the center location error and the overlap score which used to produce precision plots and success plots respectively. To obtain the precision plot, we calculate the the average euclidean distance between the center location of the tracking results and the ground truth labels. The threshold of 20 pixels is used to rank the results. For the success plot, we compute the IoU (intersection over union) between the tracking results and the ground truth labels for each frame. The AUC (area-under-curve) is used to rank the results.

### 5.5.3 Ablation Analysis

To investigate the contributions of each module and the optimal strategies to combine representations, we perform an ablation analysis with several variations of our method.

**A proper combination of features is better than features from single layer** As illustrated in Table 5.1, we experimented using features from a single layer as the target representation and combined features from several layers with different combination strategies on the two CNN models. The results show that, taken separately, conv3, conv4, conv5 give similar results. Since object appearance changes, conv3 that should give the most precise location does not always achieve good performance. However, with a proper combination, the representation power of the combined feature gets much improved.

**Features get enhanced with recalibration** Due to the Squeeze-and-Excitation operations, recalibrated features achieves better performance than the preliminary features. Recalibration through SE-block thus improves the representation power of features from single layer, which results in a better representation of the combined features.

Table 5.1 Experiments with several variations of our method, where A and S denote using the AlexNet as the basic model or using the SiamFC as the basic model.

| | | | | | | OTB-2013 | | OTB-50 | | OTB-100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Conv3 | Conv4 | Conv5 | Fusion | SE | AUC | Prec. | AUC | Prec. | AUC | Prec. |
| A | ✓ | | | | | 0.587 | 0.740 | 0.474 | 0.618 | 0.559 | 0.712 |
| A | ✓ | | | | ✓ | 0.603 | 0.755 | 0.504 | 0.642 | 0.587 | 0.747 |
| A | | ✓ | | | | 0.632 | 0.789 | 0.536 | 0.692 | 0.614 | 0.778 |
| A | | ✓ | | | ✓ | **0.637** | 0.801 | 0.544 | 0.707 | 0.623 | 0.795 |
| A | | | ✓ | | | 0.582 | 0.763 | 0.496 | 0.665 | 0.557 | 0.735 |
| A | | | ✓ | | ✓ | 0.573 | 0.762 | 0.507 | 0.696 | 0.575 | 0.769 |
| A | ✓ | ✓ | ✓ | HW | | 0.623 | 0.774 | 0.515 | 0.657 | 0.605 | 0.763 |
| A | ✓ | ✓ | ✓ | SM | | 0.633 | 0.797 | 0.542 | 0.705 | 0.616 | 0.784 |
| A | ✓ | ✓ | ✓ | SW | | 0.630 | 0.795 | 0.538 | 0.699 | 0.616 | 0.786 |
| A | ✓ | ✓ | ✓ | HW | ✓ | 0.627 | 0.798 | 0.537 | 0.700 | 0.617 | 0.790 |
| A | ✓ | ✓ | ✓ | SM | ✓ | 0.631 | 0.799 | 0.542 | 0.706 | 0.621 | 0.792 |
| A | ✓ | ✓ | ✓ | SW | ✓ | 0.635 | **0.811** | **0.545** | **0.716** | **0.627** | **0.803** |
| S | ✓ | | | | | 0.510 | 0.661 | 0.439 | 0.574 | 0.512 | 0.656 |
| S | ✓ | | | | ✓ | 0.545 | 0.709 | 0.465 | 0.608 | 0.532 | 0.687 |
| S | | ✓ | | | | 0.584 | 0.757 | 0.507 | 0.666 | 0.570 | 0.742 |
| S | | ✓ | | | ✓ | 0.592 | 0.772 | 0.518 | 0.686 | 0.581 | 0.758 |
| S | | | ✓ | | | 0.600 | 0.791 | 0.519 | 0.698 | 0.586 | 0.766 |
| S | | | ✓ | | ✓ | 0.606 | 0.801 | 0.535 | **0.722** | 0.588 | 0.777 |
| S | ✓ | ✓ | ✓ | HW | | 0.614 | 0.794 | 0.532 | 0.692 | 0.602 | 0.776 |
| S | ✓ | ✓ | ✓ | SM | | 0.612 | 0.787 | 0.539 | 0.697 | 0.607 | 0.777 |
| S | ✓ | ✓ | ✓ | SW | | 0.615 | 0.808 | 0.534 | 0.705 | 0.600 | 0.780 |
| S | ✓ | ✓ | ✓ | HW | ✓ | **0.627** | **0.823** | **0.542** | 0.716 | **0.606** | **0.787** |
| S | ✓ | ✓ | ✓ | SM | ✓ | 0.591 | 0.761 | 0.501 | 0.649 | 0.575 | 0.736 |
| S | ✓ | ✓ | ✓ | SW | ✓ | 0.603 | 0.780 | 0.518 | 0.673 | 0.590 | 0.759 |

**Multiple models are better than a single model** Our approach utilizes two CNN models as feature extractors. Here we also conducted experiments to verify the benefit of using two CNN models. As illustrated in Table 5.2, we evaluate the performance of using one CNN model and using the combination of two CNN models. The results show that the combination of two models is more discriminative than only one model regardless of the use SE-blocks.

**A proper strategy is important for the response maps combination** We applied three strategies to combine the response maps: hard weight (HW), soft mean (SM) and soft weight (SW). Since the two CNN models we used are trained for different tasks and features from different layers embed different level of information, different types of combination strategies should be applied to make the best use of features. The experimental results show that generally, combined features are more discriminative than independent features,

Table 5.2 Experiments on combining response maps from the two CNN models. $A_{conv5}$ is only taking features from the last convolutional layer of AlexNet network, $S_{conv5}$ is only taking features from the last convolutional layer of SiamFC network. $A_{com}$ is the combined response maps from AlexNet network by soft weight combining, $S_{com}$ is the combined response maps from SiamFC network by hard weight combining.

| | | | | OTB-2013 | | OTB-50 | | OTB-100 | |
|---|---|---|---|---|---|---|---|---|---|
| $A$ | $S$ | Fusion | SE | AUC | Prec. | AUC | Prec. | AUC | Prec. |
| $A_{conv5}$ | | | | 0.582 | 0.763 | 0.496 | 0.665 | 0.557 | 0.735 |
| $A_{com}$ | | | | 0.630 | 0.795 | 0.538 | 0.699 | 0.616 | 0.786 |
| $A_{com}$ | | | ✓ | 0.635 | 0.811 | 0.545 | 0.716 | 0.627 | 0.803 |
| | $S_{conv5}$ | | | 0.661 | 0.854 | 0.581 | 0.764 | 0.647 | 0.831 |
| | $S_{com}$ | | | 0.614 | 0.794 | 0.532 | 0.692 | 0.602 | 0.776 |
| | $S_{com}$ | | ✓ | 0.627 | 0.823 | 0.542 | 0.716 | 0.606 | 0.787 |
| $A_{com}$ | $S_{com}$ | HW | | 0.637 | 0.815 | 0.555 | 0.720 | 0.625 | 0.801 |
| $A_{com}$ | $S_{com}$ | SM | | 0.647 | 0.819 | 0.560 | 0.728 | 0.638 | 0.816 |
| $A_{com}$ | $S_{com}$ | SW | | 0.647 | 0.818 | 0.564 | 0.734 | 0.637 | 0.813 |
| $A_{com}$ | $S_{com}$ | HW | ✓ | 0.667 | 0.852 | **0.583** | 0.761 | 0.644 | 0.824 |
| $A_{com}$ | $S_{com}$ | SM | ✓ | 0.640 | 0.810 | 0.557 | 0.718 | 0.632 | 0.804 |
| $A_{com}$ | $S_{com}$ | SW | ✓ | **0.667** | **0.854** | 0.581 | **0.764** | **0.647** | **0.831** |

while a proper strategy can improve the performance significantly as illustrated in Table 5.1 and Table 5.2. In addition, we observe that the soft weight strategy is generally the most appropriate, except for combining hierarchical features from the SiamFC model.

### 5.5.4 Comparisons

We compare our tracker MFST with MBST [50], LMCF [47], CFNet [9], SiamFC [8], Staple [42], Struck [44], MUSTER [48], LCT [43], MEEM [45] on OTB benchmarks [1, 13]. The precision plot and success plot are shown in Fig. 5.3. Both plots show that our tracker MFST achieves the best performance among these recent state-of-the-art trackers on OTB benchmarks, except on the OTB-50 benchmark precision plot. The feature calibration mechanism we employed is beneficial for tracking as well. It demonstrates that by using the combined features, the target representation of our method is more efficient and robust. From the results, although we use siamese networks to address the tracking problem as SiamFC, and take SiamFC as one of our feature extractor, our tracker achieves much improved performance over SiamFC. Besides, despite the fact that MBST tracker employs diverse feature representations from many CNN models, our tracker achieves better results with only two CNN models, in terms of both tracking accuracy and speed.

## 5.6 Conclusion

In this paper, we presented a Multiple Features-Siamese Tracker (MFST) that exploits diverse feature hierarchies within the Siamese framework. We utilize features from different hierarchical levels and from different models using three combination strategies. Based on the feature combination, different levels of abstraction of the target are encoded into a fused feature representation. Moreover, the tracker greatly benefits from the new feature representation due to our calibration mechanism applied to different channels to recalibrate features. As a result, MFST achieved strong performance with respect to recent state-of-the-art trackers on OTB benchmarks.

## 5.7 Acknowledgments

This work was supported by The Fonds de recherche du Québec - Nature et technologies (FRQNT) and Mitacs. We also thank Nvidia for providing us with the Nvidia TITAN X GPU.

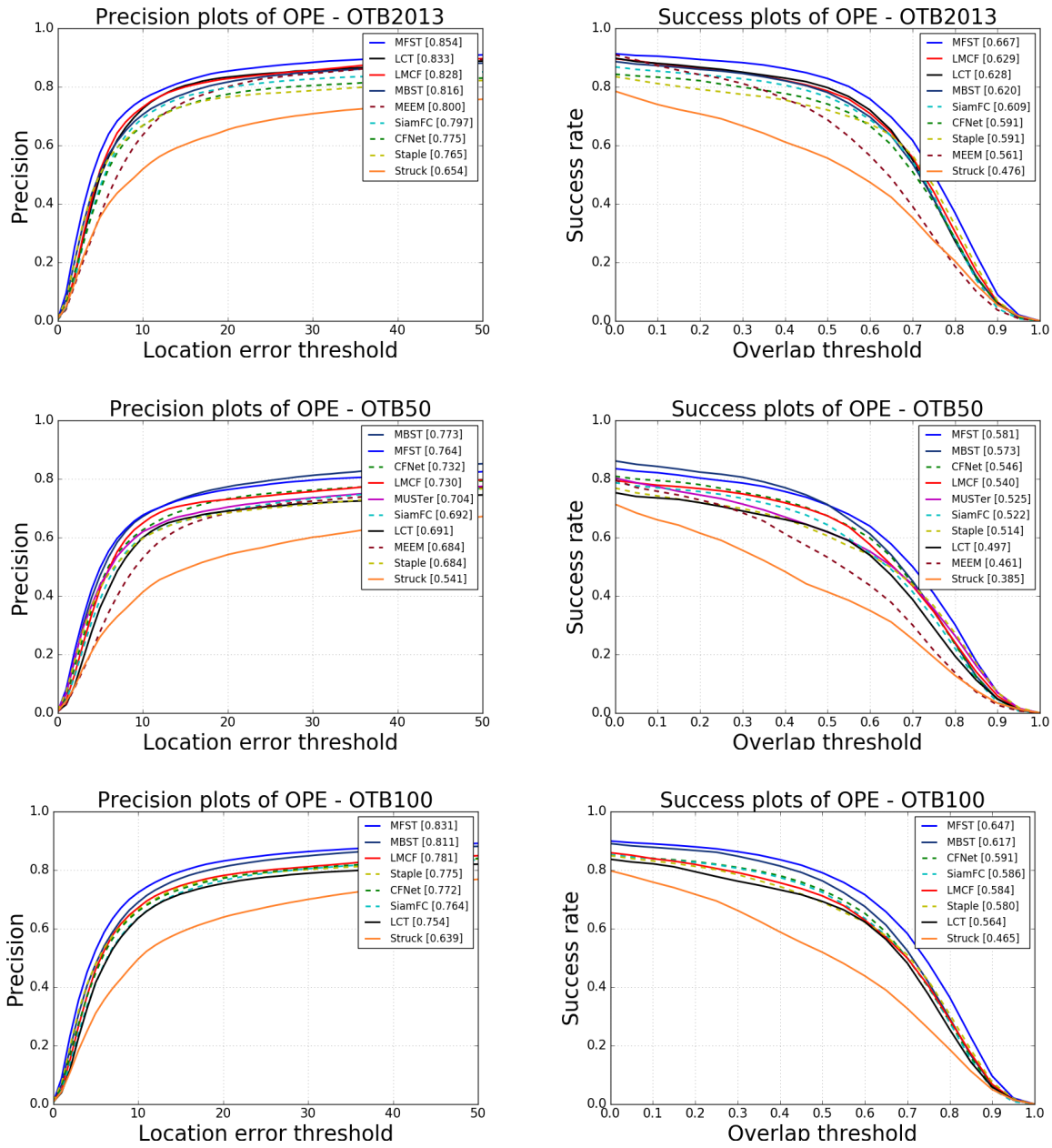



Figure 5.3 The evaluation results on OTB benchmarks. The plots are generated by the Python implemented OTB toolkit.

# CHAPTER 6 GENERAL DISCUSSION

This chapter presents a general discussion of our work. We discuss how our research progressed to achieve our objectives and present other explored directions.

**Multiple CNN models diversify the feature representation of tracking** Our work started from investigating different deep learning-based trackers. Compared with the regression network-based trackers, the fully CNN architecture of siamese trackers allows them to make full use of training data and be more discriminative. Besides, in visual object tracking, the balanced accuracy and speed achieved by siamese trackers has an important role in modern applications. Thus, we used the siamese networks as the main architecture of our models. On the other hand, most deep learning-based trackers are pretrained offline and are deployed with a single CNN model which make it difficult for them to adapt to the appearance changes of the target. It is also well known that training model online is time-consuming. Therefore, we proposed to employ multiple models in a siamese architecture. These models are constructed by different strategies with respect to the appearance categories and model diversity. Considering that the target appearance could change significantly at any time, features from multiple models in our method represent the target from different aspects. Some of them are robust to illumination variation, while others can handle rotations.

Since feature extraction is time-consuming, it is inefficient to extract features from all these models for every frame. To make full use of these models, we designed the online selection mechanism which selects branches according to their discriminative power. We investigated different ways to measure the discriminative power, including counting the number of response peak, calculating the normalized peak value of the response maps and the response sum of the peak area. The experiment showed that the best way to measure the discriminative power is calculating the distance between the peak value and minimum value of the weighted response maps. Based on the online selection mechanism, the process time gets much reduced and the best feature representation is selected.

**Combination of hierarchical convolutional features embeds more discriminative information of the target** The idea of multi-branch improves the discriminative power by employing diverse feature representations against the appearance changes. To further improve the representation ability, we exploited hierarchical convolutional features since different levels of abstraction of the target are embedded in different convolutional layers. More specifically, we took the output of the last three layers of five layers fully CNNs. We ignored

the outputs of the first two layers since they are limited by the size of the receptive field, which makes them unable to extract enough information to describe the target.

From conv3 to conv5, the receptive field increases gradually so that not only fine-grain spatial details but also semantic information can be obtained in these feature representations. Hence, we considered the ideas of multi-branch and hierarchical features together and proposed a new feature representation with a proper feature combination strategy achieving a further improved performance. Besides, the proposed method also got benefit from a feature recalibration mechanism which reweights the features for the tracking.

**A proper design is needed to apply visual attention in tracking** Different designs of visual attention mechanism can play different roles in tracking. Some of them can reweight the feature maps, making trackers more focused on the target region, while others can capture long-range dependencies in the feature maps. During our research, we attempted to apply visual attention in our methods. We first tried to learn attention maps during offline training to reweight feature maps. But since the position of the target is unknown, the offline learned attention maps cannot provide accurate weights for the target, which may mislead the tracking. On the other hand, we applied non-local neural networks after the fully convolutional neural networks to capture long-range dependencies of deep features. The experiments demonstrated that both ideas for applying visual attention cannot improve the discriminative power of our methods.

# CHAPTER 7 CONCLUSION

In this chapter, we summarize our work and present fulfillment of objectives and limitations of our algorithms. Based on what we observed and recent progress in this field, we also discuss some ideas that may advance our work in the future.

## 7.1 Summary of Works and Contributions

Our thesis focuses on visual object tracking in video surveillance. More specifically, our goal is to improve the robustness of feature representation used in tracking algorithms. Starting from introducing the background of visual object tracking, we discussed challenges and our objectives in Chapter 1. Then, we presented recent works in visual object tracking in Chapter 2, like KCF [5] and SiamFC [8]. We also presented some state-of-the-art works based on correlation filters and deep learning and standard evaluation metrics. In Chapter 3, we provided an overview of our algorithms, including MBST [50] and MFST. Chapter 4 and Chapter 5 presented our contributions. The MBST [50] presented in Chapter 4, proposed an architecture with multiple branches and online selection mechanism, showing improved representation power over the baseline it used. The MFST presented in Chapter 5, explored the effectiveness of the combination of feature maps from different convolutional layers and different CNN models with a feature recalibration mechanism.

## 7.2 Fulfillment of Objectives

Our main objective of this thesis is to investigate a robust and efficient feature representation for visual object tracking. More specifically, our goal is to: 1) discover a robust and efficient feature representation for visual object tracking, 2) develop a feature combination strategy, 3) develop a model update strategy for problems of model drifting and scale variations.

To discover a robust and efficient feature representation, we tried the following strategies: 1) assembling multiple CNNs models to extract deep features, and using online selection to select the optimal target representation, 2) fusing response maps from different CNNs models to complement each other, 3) fusing response maps from different convolutional layers of the CNNs model to integrate semantic features and features encoding fine-grained spatial details. Since a single model cannot guarantee to provide robust feature representations, there are many failure cases when only a single model is employed. However, as we are adding more branches into our architecture, the optimal representation can be selected when difficult

cases occur, which leads to better tracking performance. In Figure 4.5, the results show that MBST [50] assembling multiple feature representations achieves better performance in challenging situations. Since AlexNet [23] is trained for the image classification task, the output features from deeper convolutional layers contain more semantic information and are more invariant to object appearance changes. Since SiamFC [8] is trained to address general similarity learning problem, these semantic features extracted by AlexNet are complementary to the features extracted by SiamFC. In this manner, the fusion of features from AlexNet and SiamFC achieved improved performance as shown in Table 5.1, 5.2. As deep features and shallow features are complementary, fusing deeper and shallow layer features is able to enhance the robustness and effectiveness as well. As discussed in Chapter 5, features from shallower layers typically generate accurate predictions but is short of semantics, which leads to difficulties, when the target has significant appearance changes. Deeper features generally are invariant to appearance changes but lack details. Thus, a combination of deeper features and shallower features shows enhanced performance as illustrated in Table 5.1.

For the feature combination strategy, we tried three paradigms: 1) soft mean, normalizing each feature map by its maximum value and fusing them, 2) hard weight, assigning weight parameters for each feature map, 3) soft weight, using both normalization and assigned weight parameters. The experiment results are shown in Table 5.1.

To address the model drifting problem, we tried updating the appearance model every frame and keeping the first frame as the appearance model all the time. The experiment results show that updating the template model every frame does not improve the performances but increases the processing time. Thus, we keep the first frame as the appearance model in our trackers. Besides, in MBST, the multiple branches and the dynamic online branch selection are also beneficial for solving the model drifting problem since the most discriminative representation model is selected.

For the scale variations, we search for the object over three scales. The corresponding scale of the maximum response is considered as the best scale and is used to adjust the bounding box for tracking.

### 7.3 Limitations of the proposed approaches

From the experimental results, we observed some limitations in the methods we proposed. The first one is the accuracy of the bounding box. We scale the input images and adjust the bounding box according to the best scale, in which the maximum response value is obtained. However, we cannot scale the input images by any scale factor, as the scale factor we used

is fixed. Thus, the ratio of height to width of the bounding boxes is always the same which is entirely different from the real situation. Another limitation is the discrimination ability. The backgrounds of the training data we used are usually non-semantic, and they do not contain real object to distract the tracking. Since the siamese network we used is for learning a similarity function, it works well in discriminating foreground from the non-semantic backgrounds. When there are semantic distractors in the background, its performance is more limited. For example, the methods can work well in tracking a human body in an empty room, but it is difficult to track a human body on a busy city sidewalk. In addition, since multiple CNN models are employed to extract features, the tracking speed is affected. There is a trade-off between the tracking performance, which is related to the number of models, and the tracking speed.

## 7.4 Future Research

**Bounding box regression** As we mentioned above, the aspect ratio of bounding boxes in our methods is fixed, which limits the tracking accuracy. Based on this, our future work could focus on bounding box regression. Using regression networks, the tracking result can be more flexible. We thus suggest investigating the appropriate way to integrate regression networks in our siamese architecture.

**Distractor-aware mechanism** The siamese networks we used work well only when the background is non-semantic. The next step to improve our work would be to make our algorithm distractor-aware. To solve this problem, we cannot only increase the background diversity of the training data, but we also need to propose a distractor-aware mechanism. A distractor-aware target representation would significantly advance the discriminative power of visual object tracking algorithms.

**The trade-off between precision and speed** Although the use of multiple CNN models can diversify the feature representations, extracting multiple feature representations is time-consuming. However, decreasing the number of features may lead to performance reduction. The optimal trade-off between precision and speed remains an open VOT problem to explore.

# BIBLIOGRAPHY

[1] Y. Wu, J. Lim, and M. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[3] Z. Zhu *et al.*, "Uct: Learning unified convolutional networks for real-time visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, Oct 2017.

[4] A. Adam and C. Ioannidis, "Automatic road sign detecion and classification based on support vector machines and hog descriptos," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5, pp. 1–7, 05 2014.

[5] J. F. Henriques *et al.*, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, March 2015.

[6] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1834–1848, April 2016.

[7] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Computer Vision – ECCV 2016*, B. Leibe *et al.*, Eds. Cham: Springer International Publishing, 2016, pp. 749–765.

[8] L. Bertinetto *et al.*, "Fully-convolutional siamese networks for object tracking," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 850–865.

[9] J. Valmadre *et al.*, "End-to-end representation learning for correlation filter based tracking," 07 2017, pp. 5000–5008.

[10] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 55–73, Jan 1990.

[11] C. Ma *et al.*, "Robust visual tracking via hierarchical convolutional features," *IEEE Transcations on Pattern Analysis and Machine Intelligence*, 2018.

[12] Q. Guo *et al.*, "Learning dynamic siamese network for visual object tracking," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 1781–1789.

[13] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[14] J. Choi *et al.*, "Context-aware Deep Feature Compression for High-speed Visual Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 479–488.

[15] G. Bhat *et al.*, "Unveiling the power of deep tracking," in *ECCV*, 2018.

[16] D. S. Bolme *et al.*, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 2544–2550.

[17] M. Danelljan *et al.*, "Learning spatially regularized correlation filters for visual tracking," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 4310–4318.

[18] ——, "Eco: Efficient convolution operators for tracking," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6931–6939, 2017.

[19] B. Uzkent and Y. Seo, "Enkcf: Ensemble of kernelized correlation filters for high-speed object tracking," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2018, pp. 1133–1141.

[20] A. Bibi and B. Ghanem, "Multi-template scale-adaptive kernelized correlation filters," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Dec 2015, pp. 613–620.

[21] C. Ma *et al.*, "Hierarchical convolutional features for visual tracking," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 3074–3082.

[22] B. Li *et al.*, "High performance visual tracking with siamese region proposal network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira *et al.*, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[24] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[25] K. Chatfield *et al.*, "Return of the devil in the details: Delving deep into convolutional nets," *CoRR*, vol. abs/1405.3531, 2014.

[26] K. He *et al.*, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[28] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355

[29] B. Berlin and P. Kay, *Basic Color Terms: Their Universality and Evolution*, ser. Anthropology, linguistics, psychology. University of California Press, 1991. [Online]. Available: https://books.google.ca/books?id=sGDxruwl9OkC

[30] J. van de Weijer *et al.*, "Learning color names for real-world applications," *Trans. Img. Proc.*, vol. 18, no. 7, pp. 1512–1523, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1109/TIP.2009.2019809

[31] M. Danelljan *et al.*, "Adaptive color attributes for real-time visual tracking," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1090–1097.

[32] J. Henriques *et al.*, "Exploiting the circulant structure of tracking-by-detection with kernels," vol. 7575, 10 2012, pp. 702–715.

[33] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[34] A. He *et al.*, "A twofold siamese network for real-time object tracking," in *CVPR*, 2018.

[35] Z. Zhu *et al.*, "Distractor-aware siamese networks for visual object tracking," in *European Conference on Computer Vision*, 2018.

[36] H. Zhang *et al.*, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[37] X. Wang *et al.*, "Non-local neural networks," *CVPR*, 2018.

[38] S. Pu *et al.*, "Deep attentive tracking via reciprocative learning," in *Advances in Neural Information Processing Systems 31*, S. Bengio *et al.*, Eds. Curran Associates, Inc., 2018, pp. 1935–1945. [Online]. Available: http://papers.nips.cc/paper/7463-deep-attentive-tracking-via-reciprocative-learning.pdf

[39] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," 10 2017, pp. 105–114.

[40] B. Han, J. Sim, and H. Adam, "Branchout: Regularization for online ensemble tracking with convolutional neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 521–530.

[41] H. Nam, M. Baek, and B. Han, "Modeling and propagating cnns in a tree structure for visual tracking," *CoRR*, vol. abs/1608.07242, 2016. [Online]. Available: http://arxiv.org/abs/1608.07242

[42] L. Bertinetto *et al.*, "Staple: Complementary learners for real-time tracking," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1401–1409, 2016.

[43] C. Ma *et al.*, "Long-term correlation tracking," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5388–5396, 2015.

[44] S. Hare *et al.*, "Struck: Structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, Oct 2016.

[45] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2014.

[46] W. Zhong, H. Lu, and M. Yang, "Robust object tracking via sparsity-based collaborative model," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 1838–1845.

[47] M. Wang, Y. Liu, and Z. Huang, "Large margin object tracking with circulant feature maps," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4800–4808.

[48] Z. Hong *et al.*, "Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 749–758.

[49] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2011.239

[50] Z. Li, G.-A. Bilodeau, and W. Bouachir, *Multi-branch Siamese Networks with Online Selection for Object Tracking: 13th International Symposium, ISVC 2018, Las Vegas, NV, USA, November 19 – 21, 2018, Proceedings*, 11 2018, pp. 309–319.

[51] P. Li *et al.*, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[52] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," 2018.

[53] M. Everingham *et al.*, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[54] M. Kristan *et al.*, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, Nov 2016.

[55] ——, "The sixth visual object tracking vot2018 challenge results," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 3–53.

[56] L. Čehovin, A. Leonardis, and M. Kristan, "Robust visual tracking using template anchors," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016, pp. 1–8.

[57] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250 – 258, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865514001056

[58] F. Battistone, A. Petrosino, and V. Santopietro, "Watch out: Embedded video tracking with bst for unmanned aerial vehicles," *Journal of Signal Processing Systems*, vol. 90, no. 6, pp. 891–900, Jun 2018. [Online]. Available: https://doi.org/10.1007/s11265-017-1279-x

[59] A. Lukežič *et al.*, "Discriminative correlation filter tracker with channel and spatial reliability," *International Journal of Computer Vision*, vol. 126, no. 7, pp. 671–688, Jul 2018. [Online]. Available: https://doi.org/10.1007/s11263-017-1061-3

[60] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, June 2006, pp. 798–805.

[61] D. A. Ross *et al.*, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 125–141, May 2008. [Online]. Available: https://doi.org/10.1007/s11263-007-0075-7

# APPENDIX A SUPPLEMENTARY DETAILS OF ARTICLE 1

## A.1 Motivation and Discussion

As the deep learning technologies are widely used in many computer vision tasks, similarity learning based methods and regression networks based methods are put forward to advance this field. In our method, we take the pioneering similarity learning based approach SiamFC as our basic model due to its high tracking performance and speed. To complement the feature representation, we used AlexNet which shows satisfactory performance when it is applied in the siamese architecture for object tracking. In addition, we also explored employing VGG [2] for object tracking. We observed that its performance on most video sequences is usually lower than SiamFC and AlexNet. Adding VGG model into our architecture with some adaptations does not improve our performance. Besides, utilizing other learning models in our method needs more adaptation to fit the size of the input and output, so we used SiamFC and AlexNet as our basic models.

## A.2 Contextual Classification for Context-dependent branches

The $N_c$ context-dependent branches are trained on the $N_c$ context-dependent classes on ImageNet [24]. We do not use the class labels identified in ImageNet [24] as illustrated in A. Since ImageNet contains more than one million annotated frames, it is impractical to cluster all these frames directly. We train a context-aware network on the VOC2012 [53] dataset to group the frames of ImageNet into 10 categories. The context-aware network consists of two convolutional layers and three fully connected layers. The two convolutional layers are directly cloned from a pretrained VGG-M model [25], and the output dimensions of the three fully connected layers are 4096, 1024, 10. To train the network, there are four steps including low level feature generation, feature dimension reduction, feature grouping, and training.

**Low level feature generation** Since features from earlier convolution layers contain more spatial details, we take the outputs of the second convolutional layer of the pretrained VGG-M model as the low level feature representation of the frames of the VOC dataset, indicated by $Set\{f_{VOC}\}$. The inputs are the region of interest objects which are cropped and scaled to the dimension of $255 \times 255$, and the dimension of the outputs features are $30 \times 30 \times 256$.

**Feature dimension reduction** After that, we apply PCA on these feature maps reducing their dimensions from $30 \times 30 \times 256$ to $32 \times 900$ so that redundancies are removed before the clustering. To reduce their dimensions, we first reshape the feature maps to the size of $256 \times 900$. Then, we utilize PCA to reduce the first dimension from 256 to 32. The compressed feature maps are indicated by $Set\{f'_{VOC}\}$.

**Feature grouping and training** The next step is performing k-means on $Set\{f'_{VOC}\}$ to group them into 10 clusters. The clustering results on VOC dataset are as illustrated in Figure A.1. After we obtain the clustering labels of the VOC dataset $Set\{l_{VOC}\}$, we take the labels and the corresponding cropped images to train the context-aware network.

Finally, we use the context-aware network to classify the ImageNet to get the context-dependent subdatasets which are then used to train each context-dependent branch.

### A.3 Tracking

**Input Data Processing** We first initialize the MBST tracker with the initial frame $I_0$ and the coordinates of the bounding box of the target object which is denoted by $b_0\{x_0, y_0, w_0, h_0\}$. As we mentioned in 4.5.1, the size of the target path $z$ is $127 \times 127$ and the size of the search region $x$ is $255 \times 255$. To make the inputs compatible with our siamese network, we scale the initial frame such that the bounding box plus an added margin has area A, which can be formulated as:

$$s(w + 0.5(w + h)) \times s(h + 0.5(w + h)) = A \tag{A.1}$$

where $s$ is the scale factor and $A = 127^2$ is the scaled box centered on the corresponding position of $(x_0, y_0)$. Then, we use the area of the exemplar image as input, generate the exemplar feature maps of each branch, which are denoted as $f^0_{B_i}(i = 0, 1, ..., 11)$. For the following frames $I_n(n = 1, 3, 4...)$, we scale the images with the same scale factor $s$ and crop the images based on the last center position of the target object (for $I_1$, it is the center position of the initial bounding box). Since inputs are scaled into the fixed sizes $127 \times 127$ for the initial target object and $255 \times 255$ for search regions, video sequences with any sizes can be fed into the tracker.

**Correlation Operation** Correlation operation is used in our methods to measure the similarity between the exemplar feature map and the candidate feature maps. The search region $X$ can be considered as a collection of candidate patches $x$ in the search region with

the same dimension as $z$. Thus, for each branch, the correlation operation is implemented by taking the exemplar feature map $f^0_{B_i}$ as a filter and correlating the filter over a search window in the feature map of the search region. The output similarity scores are collected in the response map of each branch.

**Branch Selection and Prediction** Once the second frame $I_1$ is fed into the tracker, these multiple branches generate the feature maps of the scaled and cropped search region, $f^1_{B_i}$. Then, we apply cross-correlation operation with the filter $f^0_{B_i}$ in $f^1_{B_i}$ to produce the response maps $h_{B_i}$. Since the ranges of feature values from different branches are different, the value ranges of the output response maps are different, we measure their discriminative ability by Equation 4.3 to select the optimal branch $B^*$ containing the maximum value as mentioned in 4.4.2. The new position of the target object is predicted by locating the corresponding position of the maximum value in the output response map of the optimal branch. After the optimal branch is selected, only the optimal branch is activated for the following $T-1$ frames to avoid unnecessary branch selections. The correspond exemplar feature map $f_{B^*}$ is used as the correlation filter within these frames.

**Multiple Scale Searching** To handle the scale variations, instead of only cropping the original search region, we search for the target object over three scales $1.025^{\{-1,0,1\}}$. So, we also crop another two search regions based on the same center point but with larger and smaller size. They are all rescaled to $255 \times 255$ to generate the feature maps. The corresponding scale which obtains the maximum value in its response map is used to adjust the bounding box as the new bounding box.

## A.4 Hyperparameters

**Number of Contextual Clusters** TRACA [14] applies context-aware feature compression for object tracking. It is shown in their work that the optimal value is 10 for the number of contextual cluster for the ImageNet, which is validated with the OTB2013 benchmarks [13] as shown in Figure A.2. We can notice that the tracking performance decreases when the number of contextual cluster is less than 10, since the contextual variety of the ImageNet is ignored. The tracking performance also decreases when the number of contextual cluster is more than 10, which is caused by lacking training data for each category. In addition, since the number of categories identified in ImageNet is larger, and objects from the categories do not always have the same texture which is important in tracking, we do not take the original categories from ImageNet.

**Branch Selection Interval** As we mentioned in 4.5.3, the optimal value of branch selection interval $T$ is 7. When $T$ is less than 7, the tracking performance is reduced. When $T$ is more than 7, the tracking performance is decreased as well. As we know that the appearance of the target object typically changes smoothly between frames, which means that within a frame sequence, the target object keeps a similar appearance while the appearance is different compared to another frame sequence. Thus, we do not have to change the branch at each frame but using the same branch for a long time may reduce the tracking performance as illustrated in Figure 4.3.

Figure A.1 The clustering results on VOC dataset, each row is the top 10 results of each cluster.

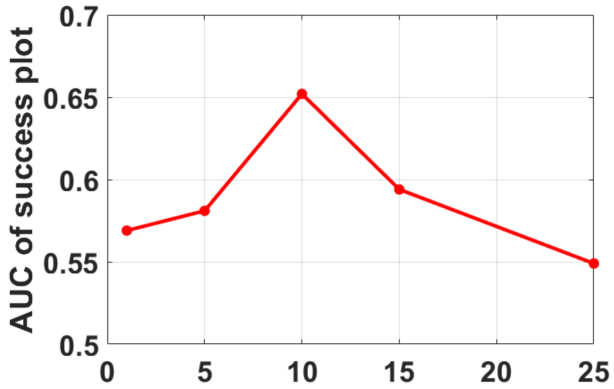
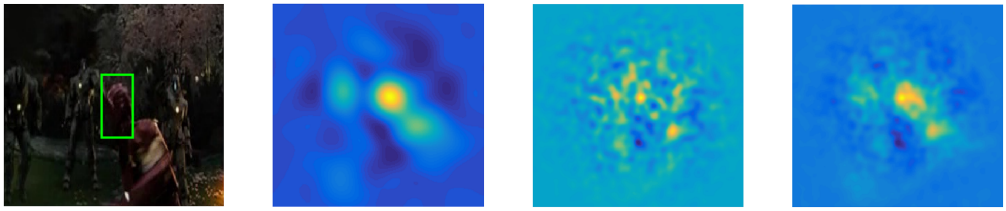



Figure A.2 Tracking performance of TRACA [14] with different numbers of the contextual clusters on OTB2013 benchmark [13]. (This image is adapted [reprinted] from [14] © 2018 IEEE)

# APPENDIX B SUPPLEMENTARY DETAILS OF ARTICLE 2

## B.1 Motivation and Discussion

In this approach, we take the pretrained SiamFC and AlexNet as our basic models since both of them show satisfactory performance in MBST [50]. As we mentioned in Article 2, different convolutional layers embed different levels of visual abstraction, features from deeper layers embed more semantic information while features from shallower layers contain more spatial details. Thus, the corresponding response maps of deep features are typically more robust, and the predictions are more reliable but coarse. On the contrary, the response maps of shallow features usually have sharp peaks providing accurate localization but less robust against distractors, as shown in Figure B.1. In this approach, instead of propagating features from earlier layers to the final layer like ResNet [26], we take pretrained models and utilize the raw hierarchical features directly as our preliminary features.



(a) Image sample (b) Deep response map (c) Shallow response map (d) Combined response map

Figure B.1 Visualization of the response maps generated by deep features, shallow features and the combined response map. (This image is adapted [reprinted] from [15] © 2018 Springer)

## B.2 Response Map Combination

We explored three different response map combination strategies in 5.4.3. Since the sizes of the exemplar feature maps are $10 \times 10 \times 384$, $8 \times 8 \times 384$ and $6 \times 6 \times 256$, and the sizes of the feature maps of the search region are $26 \times 26 \times 384$, $24 \times 24 \times 384$ and $22 \times 22 \times 256$, the correlation outputs(response maps) have the same size $17 \times 17$. We do not need to rescale these response maps for combination. In Table 5.1, we tried the three strategies, HW, SM and SM to generate the combined response maps separately, which are then used to track and evaluated on the OTB benchmarks. The optimal experimental weights of HW and SW

are obtained by these experiments. Then we choose the corresponding strategy that achieves the best performance on the OTB benchmarks to generate the combined response maps from each model, which are denoted as $r^S$ and $r^A$. After that, as illustrated in Table 5.2, we test the three different strategies again to find the best strategy to combine $r^S$ and $r^A$.

### B.3 Tracking

As for the tracking process in MBST [50], we also initialize the MFST tracker proposed in Article 2 with the initial frame and the coordinates of the bounding box of the target object. The only difference is that we do not select feature representations but need to combine response maps. After we scaled and cropped the initial frame and obtained the exemplar patch, it is fed into the basic SiamFC model and AlexNet model to generate the preliminary feature representations $S_{l_z}$, $A_{l_z}(l = c3, c4, c5)$. Then SE-block is applied to produce the recalibrated feature maps $S^*_{l_z}$, $A^*_{l_z}$, which are then used to produce response maps for tracking the target object for all the following frames.
After the feature maps of the target object are obtained, to track the target, the next frame is fed into the tracker. The tracker crops the region centered on the last center position of the target object, generate the feature representations and output the response maps by a correlation operation with the feature maps of the target object. The corresponding position of the maximum value in the final combined response map indicates the center point of the new position of the target object and the bounding box keeps the same size unless other scales obtain higher response value, which is almost the same as A.

### B.4 VOT Results

In addition to the OTB benchmarks [1, 13], we evaluate the MFST tracker on VOT2018 benchmark [54, 55] with some recent the state of the art trackers: ANT [56], ASMS [57], BST [58], CSRDCF [59], Staple [42], SiamFC [8], FragTrack [60], IVT [61], MEEM [45], Struck [44]. The results are produced by the VOT toolkit [54] as illustrate in Table B.1.

### B.5 Tracking Speed

The tracking speed of our trackers and some recent the state-of-the-art trackers is illustrated in Table B.2.

Table B.1 Evaluation results of our trackers and some recent the state of the art trackers on VOT2018 benchmark. Red: best, blue: second best, green: third best.

| | **baseline** | | | | **unsupervised** | | **realtime** |
|---|---|---|---|---|---|---|---|
| | **A-R rank** | | **EAO** | **Speed** | **Overlap** | **Speed** | **EAO** |
| | **Overlap** | **Failures** | **EAO** | **FPS** | **AUC** | **FPS** | **EAO** |
| **ANT** | 0.4559 | 40.1593 | 0.1684 | 3.2621 | 0.2789 | 4.1372 | 0.0590 |
| **ASMS** | 0.4884 | 36.5313 | 0.1692 | 134.4992 | 0.3374 | 148.6762 | 0.1669 |
| **BST** | 0.2662 | 55.1275 | 0.1160 | 1.7453 | 0.1494 | 1.3682 | 0.0531 |
| **CSRDCF** | 0.4846 | 23.5731 | 0.2561 | 8.7522 | 0.3421 | 9.4407 | 0.0993 |
| **Staple** | 0.5244 | 44.0194 | 0.1694 | 47.0093 | 0.3346 | 54.2630 | 0.1696 |
| **SiamFC** | 0.5002 | 34.0259 | 0.1880 | 31.8890 | 0.3445 | 35.2402 | 0.1820 |
| **FragTrack** | 0.3790 | 116.2327 | 0.0681 | 2.1171 | 0.1798 | 2.0917 | 0.0678 |
| **IVT** | 0.3884 | 104.7370 | 0.0757 | 8.9206 | 0.1304 | 9.8347 | 0.0652 |
| **MEEM** | 0.4558 | 33.6046 | 0.1925 | 4.1227 | 0.3275 | 4.8773 | 0.0725 |
| **Struck** | 0.4163 | 80.3253 | 0.0966 | 16.9827 | 0.1969 | 14.8882 | 0.0927 |
| **MFST** | 0.4973 | 42.7676 | 0.2000 | 26.4975 | 0.3478 | 33.2252 | 0.2000 |

Table B.2 The tracking speed of our trackers and some recent the state of the art trackers evaluated on the OTB benchmarks.

| Tracker | MFST | MBST [50] | SiamFC [8] | Staple [42] | LCT [43] | LMCF [47] |
|---|---|---|---|---|---|---|
| Speed(fps) | 39.1 | 16.9 | 86.0 | 41.1 | 27.4 | 65.6 |
| Tracker | CFNet [9] | Struck [44] | MUSTer [48] | SCM [46] | MEEM [45] | - |
| Speed(fps) | 75.0 | 9.8 | 4.5 | 0.1 | 21.8 | - |