SIAMESE NETWORKS FOR VISUAL OBJECT TRACKING

A Degree Thesis Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Sergi Sánchez Deutsch

In partial fulfilment of the requirements for the degree in

Telematic Engineering

Carried out at the

Computer Vision Lab

Institute of Visual Computing & Human-Centered Technology

Vienna University of Technology

Supervisor: Roman Pflugfelder

Barcelona, May 2019

A la meva germana Esther, que
aconseguirà tot el que es proposi.
You can and you will.

Abstract

Visual object tracking has become one of the hottest topics in computer vision since its appearance in the 90s. It has a wide range of important applications in real life, such as autonomous driving, robot navigation and video surveillance. Despite the efforts made by the research community during the last decades, arbitrary object tracking is still, in its generality, an unsolved problem.

Recently, some tracking algorithms have used convolutional neural networks trained from large datasets, providing richer image features and achieving more accurate object tracking. Results show that deep learning techniques can be applied to enhance the tracking capabilities by learning a better model of the object's appearance. The aim of this thesis is to study and evaluate the implementation of one method of this approach called SiamFC and to give a brief overview of the current tracking challenges. The code developed in this study makes use of an existing Python implementation of SiamFC and is publicly available at https://github.com/sergi2596/pytorch-siamfc

Resum

El seguiment d'objectes s'ha convertit en un dels temes més candents en visió artificial de les últimes dècades. Es pot aplicar a multitud de situacions a la vida real, com per exemple conducció autònoma, robòtica i videovigilància. Tot i que la comunitat científica ha estat molt activa investigant en aquest camp, el seguiment d'objectes és encara un problema complex que necessita ser millorat.

Recentment, alguns algoritmes han utilitzat les xarxes neuronals convolucionals entrenades amb grans bancs de dades per oferir un seguiment d'objectes millor i més fiable. Els resultats mostren que les tècniques d'aprenentatge profund es poden aplicar per millorar les capacitats de seguiment gràcies a la oportunitat d'aprendre models més complexos de l'aparença dels objectes. L'objectiu d'aquest treball és estudiar i provar la implementació d'un d'aquests algoritmes anomenat SiamFC, així com donar una visió global dels reptes actuals del seguiment d'objectes. El codi desenvolupat en aquesta tesis està basat en una implementació ja existent del SiamFC basada en Python i està a https://github.com/sergi2596/pytorch-siamfc

Resumen

El seguimiento de objetos se ha convertido en uno de los temas más candentes en visión artificial de las últimas décadas. Se puede aplicar a multitud de situaciones en la vida real, como por ejemplo la conducción autónoma, la robótica o la videovigilancia. A pesar de que la comunidad científica ha estado investigando activamente en este campo, el seguimiento de objetos es todavía un problema complejo que necesita ser mejorado.

Recientemente, algunos algoritmos han utilizado las redes neuronales convolucionales entrenadas con grandes bancos de datos para ofrecer un seguimiento de objetos mejor y más fiable. Los resultados muestran que las técnicas de aprendizaje profundo se pueden aplicar para mejorar las capacidades de seguimiento gracias a la oportunidad de aprender modelos más complejos de la apariencia de los objetos. Este trabajo busca estudiar y probar la implementación de uno de estos algoritmos conocido como SiamFC, así como dar una visión global de los retos actuales del seguimiento de objetos. El código desarrollado en esta tesis está basado en una implementación ya existente de SiamFC basada en Python y está disponible en https://github.com/sergi2596/pytorch-siamfc.

Acknowledgements

I am deeply grateful to Roman Pflugfelder for introducing me to the world of object tracking and for guiding me throughout these months of work. Thanks for your valuable comments.

I thank my family for helping me achieve my goals and for supporting me during my academic life.

And thanks to my friends for being my dose of relief during all these days of hard work. They know who they are.

Revision history and approval record

Revision	Date	Purpose
0	08/04/2019	Document creation
1	29/04/2019	Document revision
2	10/05/2019	Document approval

Document distribution list

Name	e-mail
Sergi Sánchez Deutsch	sergi.sanchez.deutsch@estudiantat.upc.edu
Roman Pflugfelder	roman.pflugfelder@tuwien.ac.at
Xavier Giró Nieto	xavier.giro@upc.edu

Written by		Reviewed and approved by	
Date	08/04/2019	Date	10/05/2019
Name	Sergi Sánchez Deutsch	Name	Roman Pflugfelder
Position	Project Author	Position	Project Supervisor

Contents

List of figures

List of tables

1

Introduction

Visual object tracking (VOT, also referred to as object tracking) is the process of locating a moving object (or multiple objects) over time in a video. Object tracking is an important computer vision topic with a great number of useful real-world applications such as autonomous vehicles [1], robotics [2], human-computer interaction[1], security and video surveillance [3]. Since its appearance in the 90s, object tracking has become a very active research topic in Computer Vision. Although results have been significantly improved over the years, the process of tracking arbitrary objects in arbitrary scenes remains unsolved as research has not brought a breakthrough.

The ability to perform reliable and effective object tracking depends on how well trackers can deal with challenges such as occlusion, scale variations, low resolution targets, fast motion and presence of noise. Different objects have also different appearances, and rotations and deformations are problems that need to be addressed too. Moreover, the vast majority of applications require object tracking algorithms to operate in real-time, which adds an extra level of difficulty due to the time constraint.

Traditionally, the problem of tracking an object has been solved by learning a model of the target's appearance using only the frames of the video itself. This method has been proved to work well in many cases, but there are important limitations in the complexity of the model that can be learnt. The main consequence of this is that only objects with similar appearances can be correctly tracked. To address a more general problem and track arbitrary objects in arbitrary scenes, performing online learning using only data extracted from the video frames may be insufficient.

It has been proved in the last years that object tracking algorithms can take advantage of the power of deep learning techniques, i.e. using deep convolutional networks trained from large supervised datasets, to learn a richer set of features from a variety of objects. This approach is rather a new topic that can bring notorious improvements to the tracking scene.

[1] e.g. gesture recognition, eye gaze tracking for data input to computers, etc.

1.1 Purpose

The purpose of the current thesis is to analyse and evaluate the implementation of *SiamFC* [4], a tracking algorithm that merges the power of Siamese networks and deep learning to perform arbitrary object tracking. The thesis also aims to give a brief overview of the most important challenges and principles in object tracking. This work has been carried out at the Institute of Visual Computing and Human-Centered Computing of the Vienna University of Technology. It starts from scratch and it has been developed independently from other projects in the department.

1.2 Outline

The rest of the thesis is organized as follows: chapter 2 provides a brief overview of the main challenges and approaches to perform tracking, as well as an introduction to siamese networks; the description of SiamFC architecture and operation is explained in chapter 3; in chapter 4 we perform experiments and evaluate the results using variations in datasets, training and tracking parameters; and chapter 5 discusses conclusions and future work.

1.3 Work plan

1.3.1 Work packages

<table>
<tr><td>Project: Documentation</td><td colspan="2">WP ref: WP1</td></tr>
<tr><td>Major constituent: Documentation</td><td colspan="2">Sheet 1 of 5</td></tr>
<tr><td rowspan="2">Short description: Writing of the thesis documents</td><td colspan="2">Planned start date: 01.10.2018
Planned end date: 10.05.2019</td></tr>
<tr><td colspan="2">Start event: -
End event: -</td></tr>
<tr><td>Internal task T1: project proposal redaction
Internal task T2: project proposal revision
Internal task T3: project proposal approval
Internal task T4: critical review redaction
Internal task T5: critical review revision
Internal task T6: critical review approval
Internal task T7: final report redaction
Internal task T8: final report revision
Internal task T9: final report approval</td><td>Deliverables:

- Project Proposal
- Critical Review
- Final Report</td><td>Dates:

- 05/10/18
- 30/11/18
- 11/05/19</td></tr>
</table>

Table 1: Description of work package 1

Project: Literature review	WP ref: WP2	
Major constituent: Documentation	Sheet 2 of 5	
Short description: Study and understanding of Deep Learning and Visual Tracking concepts as well as SiamFC	Planned start date: 05/10/18 Planned end date: 20/12/18	
	Start event: Project Start End event: -	
Internal task T1: Object tracking literature study Internal task T2: Study of SiamFC paper	Deliverables: -	Dates: -

Table 2: Description of work package 2

Project: Software Implementation	WP ref: WP3	
Major constituent: Software	Sheet 3 of 5	
Short description: work on the implementation of SiamFC	Planned start date: 20/12/18 Planned end date: 11/03/19	
	Start event: - End event: -	
Internal task T1: Study of SiamFC original implementation in Matlab Internal task T2: learn PyTorch Internal task T3: implement a modified alexnet following SiamFC structure Internal task T4: Study and modify the python implementation of SiamFC Internal task T5: Create script for synthetic dataset Internal task T6: Create script for synthetic video	Deliverables: -	Dates: -

Table 3: Description of work package 3

Project: Experiments	WP ref: WP4	
Major constituent: Research	Sheet 4 of 5	
Short description: perform experiments using different datasets	Planned start date: 11/03/19 Planned end date: 11/04/19	
	Start event: - End event: -	
Internal task T1: Training experiments with synthetic dataset Internal task T2: Training experiments with ImageNet	Deliverables: -	Dates: -

Table 4: Description of work package 4

Project: Oral communication	WP ref: WP5	
Major constituent: Documentation	Sheet 5 of 5	
Short description: preparation of final presentation	Planned start date: 13/05/19 Planned end date: 29/05/19	
	Start event: Final Report End event: Oral Presentation	
Internal task T1: preparation of slides Internal task T2: oral defense	Deliverables:	Dates:

Table 5: Description of work package 5

1.3.2 Gantt diagram

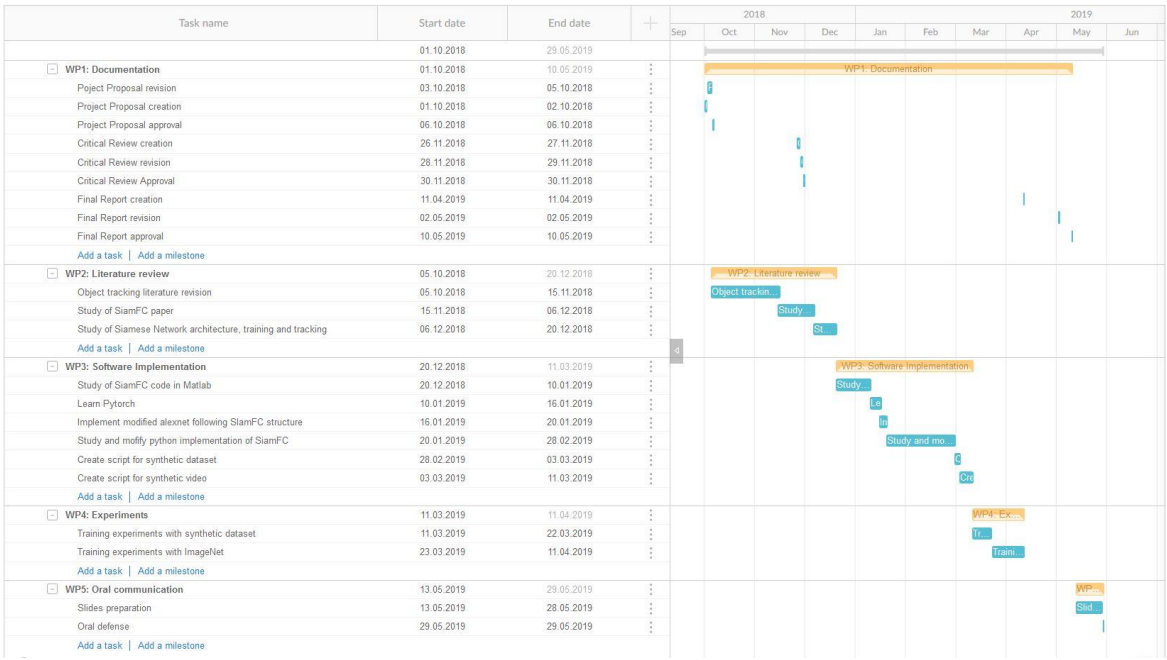

Fig. 1: Gantt diagram

2

Background

The first section of this chapter (2.1) presents the main parts of the tracking process, giving a general explanation and mentioning the differences between some common approaches. In section 2.2 we briefly mention the main challenges that tracking has to deal with. Section 2.3 gives a general overview of the principles behind learning tracking. Section 2.4 describes the main concepts behind Siamese networks and some background about SiamFC is explained in section 2.5.

2.1 The tracking process

As described in literature such as [5], [6] and [7], the process of tracking an arbitrary target in a video is usually divided into several steps: object state modelling, features and feature selection and object tracking. Since it is a very complex process, most of the algorithms simplify tracking by applying some constraints. For example, it is common to assume that the motion of the target to track is smooth and that velocity or acceleration is constant. Other common constraints are prior knowledge about the number, size and appearance of the objects.

2.1.1 Object state modelling

In a tracking scenario, an object can be represented by its shape and appearance. These two ways of representation can be used combined or alone, and the choice is determined usually by the application domain and purpose. A suitable tracking algorithm must be chosen accordingly to the object representation. This section covers the shape representations that are usually employed for tracking [5].

Points: the target is represented either by a centered point (Fig. 2 (a)) or by a set of points (Fig. 2 (b)) [8]. It is usually used for small sized objects. Using multiple points can cause trouble when tracking multiple objects that have some interaction during the video.

Geometric shapes: primitive geometric (Fig. 2 (c), (d)) shapes such as rectangles and ellipses [9] are suitable for representing simple rigid objects, although it can also be used with non-rigid objects. Since objects usually have more complex shapes, this method causes

some parts of the object to be excluded from the shape template or some parts of background to be included.

Articulated shape models: articulated objects are those composed by different parts that are grasped together with joints. These parts can be modelled using ellipses or cylinders (Fig. 2 (e)).

Skeletal models: the skeleton of the object (Fig. 2 (f)) is extracted by applying medial axis transform[2] to the object silhouette. Skeletons can be used to represent both articulated and rigid objects.

Object silhouette and contour: contour (Fig. 2 (g), (h)) defines the outline of an object and the area inside the contour is known as the silhouette (Fig. 2 (i)). This is a flexible method commonly used for representing complex non-rigid objects.

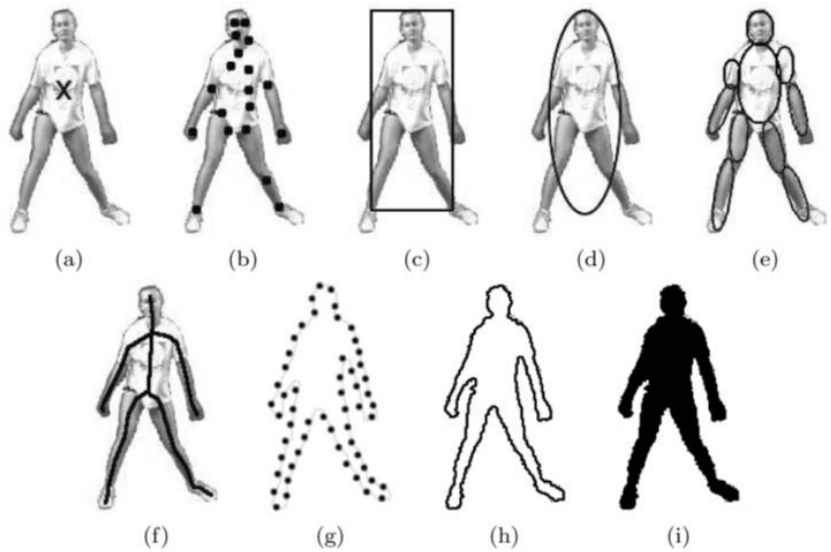

Fig. 2: Object representations. (a) centered point, (b) multiple points, (c) rectangular shape, (d) elliptical shape, (e) articulated model, (f) object skeleton, (g) (h) different object contours, (i) object silhouette. Source: [5]

2.1.2 Feature selection

In computer vision, a feature is a transformation that helps extracting relevant information for solving a certain computational task. In the case of tracking, a feature can be an "interesting" part of an image that contains useful and accessible information to distinguish the object that is being tracked. Feature selection [10], [11] is the process of selecting a subset of relevant features so any kind of redundant or irrelevant information is removed. It provides simpler models, which results in shorter training times.

[2] The **medial axis** of an object is the set of all points having more than one closest point on the object's boundary. Medial Axis Transform (MAT) is a representation that encodes an object with symmetric (medial) axes in the object interior.

Features may describe objects at different levels of detail. For example, a neural network trained for object detection can extract simple features like edges and contours in its first layers. The deeper layers may be capable of detecting more complicated patterns, such as textures, complex shapes or variations between detected features.

2.1.3 Object tracking

The tracking part of the process addresses the task of establishing a correspondence between object instances across video frames. Tracking can be performed using a broad number of inference algorithms which handle different situations. Some focus on tracking single objects while others take into account multiple targets. A tracker can also deal with partial and total occlusions of the targets. The tracking algorithms can be classified by different criteria [12]. A commonly way to classify tracking methods is by dividing it into three main categories based on the object state: point tracking, kernel tracking and silhouette tracking [5]. These categories are also divided into subgroups, but this thesis covers only the differences between the main categorization.

Point tracking: this method is used when objects are represented by points. To associate points between frames the tracker takes into account the previous state of the object, which can include information about position and motion. Some constrains are applied by assuming, for example, that the object position doesn't change drastically or that the velocity is similar between consecutive frames.

Kernel tracking: in this context, kernel refers to the shape and appearance of an object. This approach computes the motion of an object represented by a primitive object region such as rectangle of ellipse. Objects are tracked by considering the coherence of their appearances in consecutive frames.

Silhouette tracking: the goal of this approach is to find a region that matches an object model that was generated using the previous frames. This model can include information about color histogram, edges and contours of an object. Silhouette tracking provides a more accurate shape description than simple geometric shapes, and is therefore used to track objects with complex shapes.

2.2 Tracking challenges

Tracking for arbitrary objects and scenes is in its generality unsolved as there has not been an important breakthrough since its appearance in the 90s. So far, the research community has only brought partial solutions that work under rather specific constraints. The fact that tracking is such a complicated task can be briefly summarized by three important challenges [13]:

Uncertainty: trackers need to deal with a certain amount of changes during the tracking process to prevent failure. For example, an ideal tracking algorithm should be invariant to variations in the object appearance caused by deformation, rotation, scale variations,

changes in illumination or camera movement. Uncertainty refers to the risk of potential changes that are unknown and cannot be controlled by the tracker.

Initialisation: trackers are usually initialised by humans in the first frame. Automatized initialisation without human intervention, e.g. by an object detector, is crucial to re-initialise the tracker after a full occlusion of an object. This is still a huge challenge as object detectors can only detect certain object categories.

Computability: there is a need to perform accurate and efficient object tracking in order to be practical for real world applications. This balancing between accuracy and speed remains a major challenge.

2.3 Learning principles

The learning of object tracking can be performed either online or offline [13].

Online learning is a method in which the data is available in a sequential order and is used in real time by the algorithm to update the predictor for future steps [14]. It is the most used way of learning object tracking, especially to follow categorical objects such as vehicles and persons in video surveillance applications. Online learning has been seen in an unsupervised or semi-supervised way, which means that either the data is unlabelled or there is a small amount of labelled data and the algorithm extends its predictions to unlabelled data [15].

Offline learning takes a static set of input data and do not change the approximation of the target function once the training process is completed. Despite being a well-known approach in many other machine learning applications, this is rather a new topic in the tracking field that was not used until 2015. Recently, it has been exploited to create more complex models of object representations by learning richer sets of spatiotemporal features from a variety of object categories. The expectation of this approach is to improve the performance when addressing the problem of tracking arbitrary targets in arbitrary scenes.

Offline learning is usually supervised, which means that is limited by the amount of labelled data available. However, the recent appearance of large image datasets [16][17] can suppose a considerable reduction on this constraint.

2.4 Siamese networks

Siamese networks were first introduced by J. Bromley and Y. LeCun in 1994 to design a signature verification system for a pen-input tablet [18]. These networks can be really simple, yet they are getting increasingly popular and have been used in multiple applications

such as face verification [19], object cosegmentation[3] [20], one-shot[4] image recognition [21] and to detect similar questions in online Q&A platforms [22].

A siamese network consists of two twin subnetworks that extract features of two different inputs [23]. A final layer connects the output of these twin subnetworks and computes the distance between them. Siamese networks are trained to compute the similarity between two inputs and to decide whether they belong to the same class or not. Since this task is usually a binary classification, one of the most common loss functions employed is the binary cross-entropy loss

$$L = y \cdot \log(p) + (1 - y) \cdot \log(1 - p) \tag{1}$$

where y is the class label (0 or 1) and p the predicted class.

To obtain a system that is symmetric and invariant to switching the inputs, both branches of the network must share the same weights and bias. An example of a siamese network architecture can be found in Fig. 3.

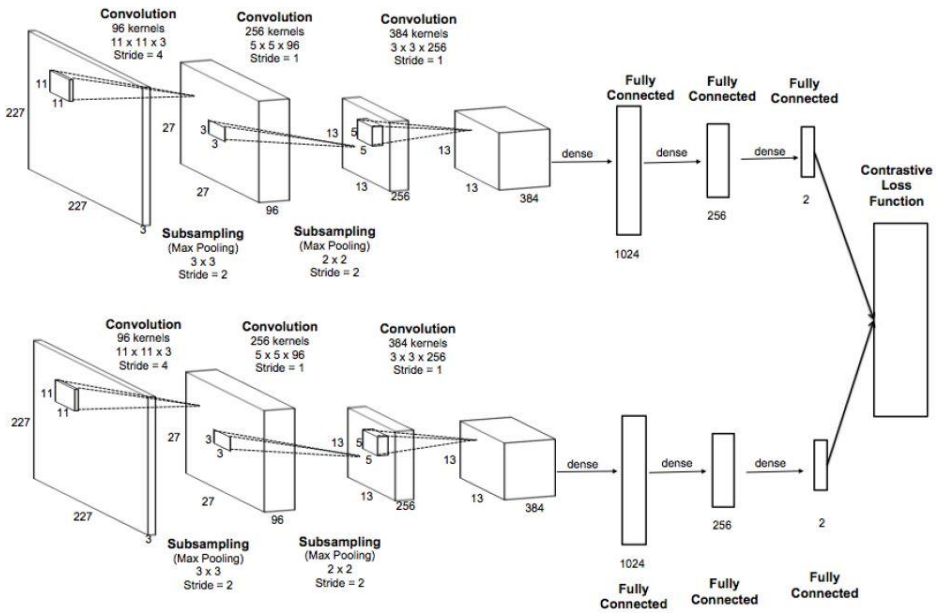

Fig. 3: Example of a Siamese Neural Network architecture used in [34] to compute the similarity between facial expressions. Note that the network is composed by two identical branches connected at the end by a loss function.

[3] **Object cosegmentation** is the process of discovering similar objects from multiple images and segmenting them as foreground simultaneously.

[4] **One-shot** learning addresses the problem of learning information about an object category from only one sample or a few training samples.

2.5 SiamFC

The main goal of this thesis is to study the tracking algorithm *SiamFC*. Despite the simplicity of its architecture, SiamFC performs very well in the most important object tracking benchmarks. In 2017, the tracker won the VOT real-time challenge [24]. It also ranks very high in some important benchmarks presented during 2018 such as LaSOT [25] and GOT-10k [16]. The LaSOT dataset is composed by 1,400 videos containing 3,52 million images with 70 different object classes. In the case of GOT-10k, the dataset contains 10,000 videos with 1.5 million images in total and 563 object categories. The good performance on these new benchmarks make SiamFC an interesting tracker to analyse and opens the door to important improvements on arbitrary object tracking.

3

SiamFC

This chapter covers the corresponding explanations of SiamFC, the tracker proposed by L. Bertinetto et. Al. [4] in 2016. It includes an overview of the tracker architecture as well as the training and tracking phases.

3.1 Introduction

The goal of SiamFC is to track an arbitrary object in a video without having any prior knowledge of it. The object is only identified in the very first frame of the video by a rectangular bounding box. The algorithm must be able to track any object so it is not feasible to train the tracker for a specific object or group of objects. Instead of learning a specific detector, the algorithm addresses a more general *similarity* problem by training a siamese network in an offline phase to compare an exemplar image z to a larger search image x and give a high value if both images contain the same object. An in-depth revision of the network architecture is done in the next section.

3.1.1 Network architecture

As shown in Fig. 4, the network is divided into two branches each one taking an image as input. Both images are frames extracted from the video itself. During tracking, the exemplar image z is a cropped version of the first frame of the video containing the initial appearance of the target to track. The rest of the video frames are passed to the network as larger search images x that are centered at the previous position of the target.

An identical transformation φ is applied to both inputs to extract a representation of the images. These representations are the feature maps of the last layer of the subnetworks. Both feature maps are then combined using a cross-correlation layer according to

$$f(z, x) = \varphi(z) * \varphi(x) + b\mathbb{1} \tag{2}$$

where $b\mathbb{1}$ is a bias with value $b \in \mathbb{R}$. Using this cross-correlation layer, the network can determine the position of the object in a new frame by exhaustively testing all possible

locations of the exemplar image z within the search image x. This property allows SiamFC to be more efficient by using less training data.

The output of the network is a scalar-valued score map representing the similarity for each of these locations. Finding the position of the object in a new image is done by choosing the candidate position with the highest value in the score map and computing the distance to the center.

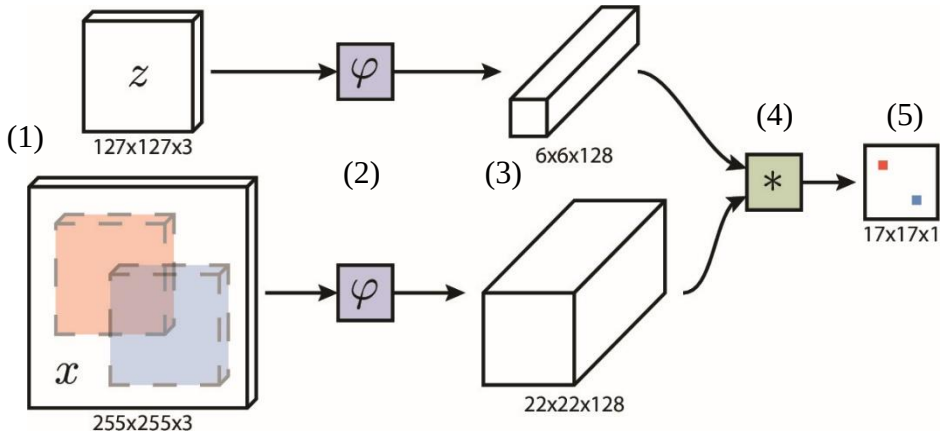

Fig. 4: SiamFC network architecture. (1) input images, (2) embedding function, (3) image feature maps, (4) cross-correlation layer, (5) resulting score map. Source: [4]

The network applies an embedding function φ to both images to extract their corresponding feature maps. This function is created using a convolutional neural network which is an adaptation of the convolutional part of the well-known neural network *AlexNet* [26] presented by A. Krizhevsky et. Al. in 2012. The network consists of five convolutional layers and two max pooling layers to introduce invariance to slight appearance changes of the target. A complete schema of this architecture is shown in Fig. 5. A ReLu follows every convolutional layer except for the last one.

An important feature of SiamFC architecture is that the network is fully convolutional with respect to the search image. In a convolutional network [27] the first layer is the image itself and locations in the following layers correspond to the locations in the original image that they are path-connected to. A convolutional network is set up with components such as convolution, pooling and activation functions whose output depends only on a local input region. The main advantage of this is that these networks obey translation invariance so the input can be of any size. This property allows the SiamFC to, instead of taking two images with the same size, provide a much larger search image as input to compute the similarity at every position using translated sub-windows.

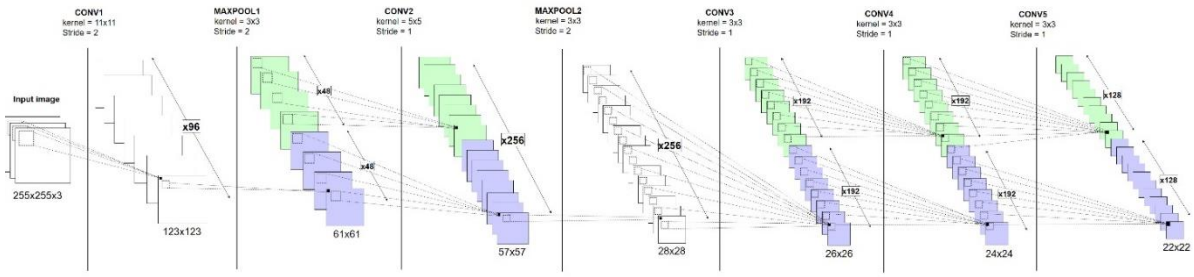

Fig. 5: Architecture of the convolutional neural network used as embedding function φ. ReLu is applied after each convolutional layer except for the last one. Note that conv1 and conv3 are normal convolutions while conv2, conv4 and conv5 are grouped convolutions. This means that input channels are split into two groups so each filter is convolved with only half of the previous feature maps.

3.1.2 Dataset curation

SiamFC was originally trained from the ILSVRC 2015 dataset for object detection in video [28] (also known as ImageNet) which contains 4417 videos gathering more than 2 million labelled images in total. The dataset curation, which is done in an offline phase before training the network, consists on creating training pairs of images to use as inputs to the network. Each pair is composed by an exemplar and a search image, both extracted from the same video and centered at the position of the target. Images are scaled preserving the original aspect ratio to get search images of 255 x 255 pixels and exemplar images of 127 x 127 pixels. If an image cannot be fit in these sizes without corrupting the aspect ratio, the portions that extend the image size are filled with the mean RGB value of the image, as shown in Fig. 6.

Fig. 6: Training pairs extracted from ImageNet dataset. Top images are 127 x 127 and bottom images are 255 x 255. Note that pairs are extracted from the same video.

3.1.3 Training

During training, pairs of search and exemplar images are applied into the network. The cross-correlation layer produces a scalar-valued score map $v: \mathcal{D} \rightarrow \mathbb{R}$ whose dimensions depends on the stride k of the cross-correlation layer. This score map represents the similarity between every position of the search image and the reference image.

The logistic loss is employed to calculate the loss for each component of the score map and the loss of the score map is defined as the mean of the individual losses

$$L(y, v) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \log(1 + e^{-y[i] \cdot v[i]}) \tag{3}$$

where $v[i]$ is the value of the score map component and $y[i]$ is the ground-truth label, which takes values $y[i] \in \{+1, -1\}$. Once the loss is computed, backpropagation [29] is applied by using Stochastic Gradient Descent for the optimization.

Since all images in the dataset have been centered on the target before training, it is known beforehand that the maximum value of the score map should be centered as well. Therefore, an element of the score map is considered to belong to the target if it is within a radius R of the center. This is reflected by a ground truth mask, which is the same for all training pairs and takes the form of a matrix with the same size as the score map

$$y[i] = \begin{cases} +1 & if\ i \leq R/k \\ -1 & otherwise \end{cases} \tag{4}$$

where k is the stride of the cross-correlation layer. Since there are more values belonging to background than to the target, the individual losses are weighted to avoid class imbalance by applying

$$w[i] = \begin{cases} \dfrac{0.5}{sumPos} & if\ y[i] = +1 \\ \dfrac{0.5}{sumNeg} & if\ y[i] = -1 \end{cases} \tag{5}$$

where $sumPos$ and $sumNeg$ are the total number of positive and negative values in the ground truth mask.

3.1.4 Tracking

The online phase of the network consists on evaluating the equation (2) for every frame of the video. The first frame, where the object is identified with a bounding box, is used as the exemplar image z for the whole video. This means that the feature map of the exemplar image is computed only once and is fixed for the rest of the tracking process. The feature map for the search image is computed for all the remaining video frames.

During the tracking process the search images are centered at the previous position of the target. After the cross-correlation layer, the new position of the target is given by the displacement between the center of the score map and the position its maximum value. A cosine window is applied to penalize large displacements, assuming that the motion of a real object must be smooth.

For each frame, the algorithm handles scale variations by searching the target over five scales. The score map is computed for each scale and the one with the highest maximum value is selected to calculate the displacement.

Before calculating the displacement of the target, the score map is upsampled from 17 x 17 to 272 x 272. The authors of SiamFC state that this technique improve the accuracy in localization.

4

Experiments

4.1 Implementation details

This section covers relevant explanations regarding the code implementation and the procedures followed during experimentation. The experiments have been performed using an existing implementation of SiamFC [30] that has been studied and modified according to the needs of the thesis. Other parts of the implementation have been developed from scratch. All the code is written in Python[5] and make use of the deep learning framework *Pytorch*[6]. It is publicly available on https://github.com/sergi2596/pytorch-siamfc and it includes a step by step description to reproduce the experiments of this thesis.

4.1.1 System setup

All the experiments have been performed using the resources provided by Vienna Scientific Cluster[7]. Depending on the task and availability, we employed either one or multiple NVIDIA GeForce GTX 1080 with 8gb of RAM each.

4.1.2 ImageNet dataset

SiamFC was originally trained using the ImageNet[8] dataset, and so we use it to perform some of the experiments. ImageNet is a large visual database for use in visual object recognition tasks that includes three main challenges to evaluate algorithms in object localization, object detection and object detection from video. Specifically, we use the dataset for object detection in video, which consists in 4417 videos containing more than 2 million labelled images with 30 different object categories.

[5] https://www.python.org/
[6] https://pytorch.org/
[7] http://vsc.ac.at/
[8] http://image-net.org/

4.1.3 LaSOT dataset

LaSOT [25] is a recently released dataset for Large-scale Single Object Tracking. It contains 1,400 videos gathering 3.52 million annotated images with 70 different object categories. We use some individual videos to evaluate the performance of the tracker in some of the experiments.

4.1.4 Synthetic dataset

Before training SiamFC on a real dataset such as ImageNet, we have created a synthetic dataset consisting on a set of 255 x 255 images, each one containing a simple centered square with a fixed size as shown in Fig. 7. Images are clustered together in subfolders.

The color of the square and background is randomly chosen for each subfolder. We also apply random noise type with random intensity level to each image. It is well known that adding noise to input when training usually improves the robustness of the network, which results in better generalization and faster training.

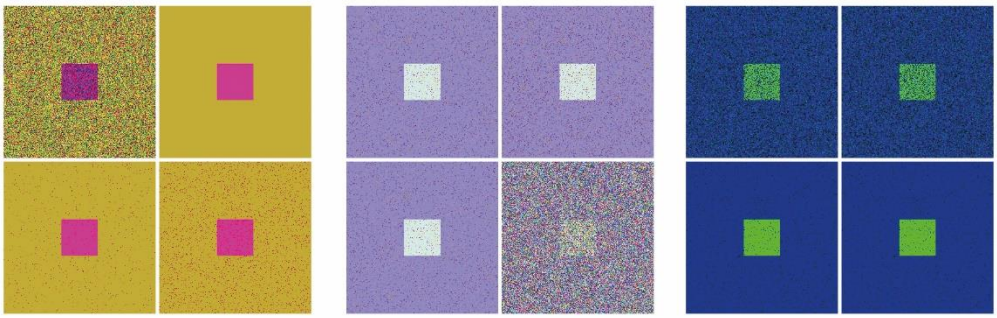

Fig. 7: Samples extracted from synthetic dataset. Note that each group belongs to a different subfolder and that images from the same group have fixed colors and random noise.

The purpose of using this dataset is to assure the proper functioning of the network when it's trained with a dataset containing a simple easily identifiable target. We also aim to compare the performance of a tracker trained with this dataset and a tracker trained with ImageNet. Due to space limitations, we have created for this thesis a dataset of 500k images grouped in 2000 subfolders, each one containing 250 samples.

4.1.5 Synthetic video

Following the idea of the previous section, we have also developed a synthetic video to test the network once is trained with the synthetic dataset. The target to track in the video is a black square with a fixed size on top of a white background, as shown in Fig. 8. The motion

of the target is random and the maximum displacement from frame to frame can be chosen when the video is created.

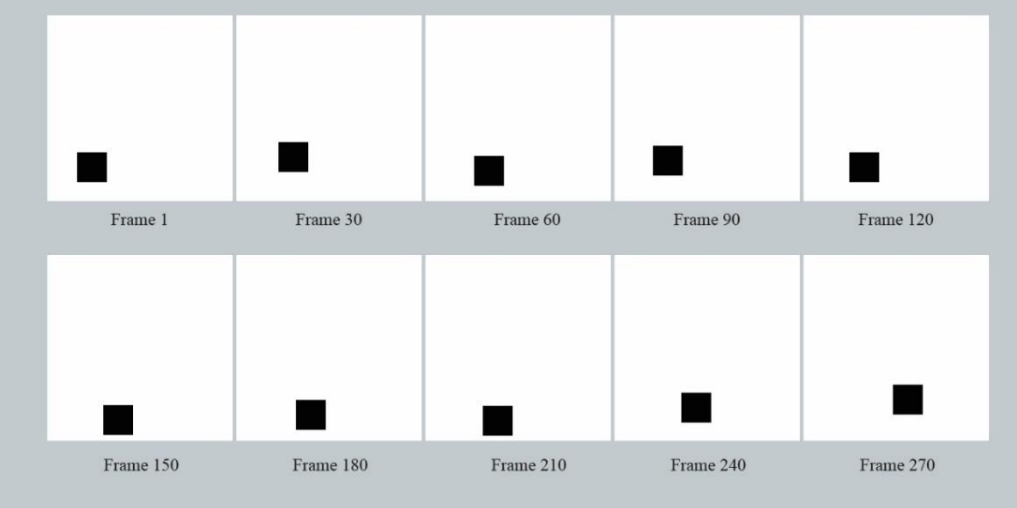

Fig. 8: Samples extracted from a synthetic video. In this example, the video contains 300 images that are 255 x 255 pixels. The target has a size of 40 x 40 pixels and the maximum displacement from frame to frame is set to 4 pixels.

4.1.6 Training

The implementation of the training process follows the procedure explained in section 3.1.3. All experiments are performed over 50 epochs using mini-batches of 8 samples. During training, we split the dataset using 90% for training and 10% for validation. The validation phase is performed after every epoch. To initialize the weights of the network, we employ a Xavier initialization [31].

Table 6 describes important training parameters that are fixed for all the experiments.

Parameter	Description	Value
exemplar_size	Size of exemplar images (in pixels)	127
search_size	Size of search images (in pixels)	255
train_batch_size	Size of batch for training	8
valid_batch_size	Size of batch for validation	8
learning_rate	Learning rate of optimizer	10^{-2}
epoch	Number of epochs	50

radius	Radius to define which positions of the score map belongs to the target and which to the background.	16
train_ratio	Ratio of training data over validation data	0.9

Table 6: Description and values of training parameters.

4.1.7 Tracking

The tracking algorithm follows the description of section 3.1.4. More specifically, the algorithm initializes the tracker with the first frame of the video containing the target with its corresponding bounding box. This first frame is used as the exemplar image for the whole tracking. We store the initial position and size of the target and then, for each of the following video frames:

1. We compute the score map for different scales of the target and upsample them using a bicubic interpolation.
2. We choose the score map with the highest maximum value and we update the size of the target.
3. We apply a cosine window to penalize large displacements of the target.
4. We choose the position in the score map with the highest value and we compute the distance to the center. The new position of the target is computed by adding the displacement to the previous position.

4.1.8 Evaluation metric

We evaluate the results of the experiments by computing the Center Error for every frame i of a given video. The center error is defined by the Euclidean distance between the center of the predicted bounding box and the center of the ground truth bounding box. We then compute the mean of the Center Error for all the frames of the video

$$center\_error = \frac{1}{N}\sum_{i}\sqrt{(\hat{x}[i]-x[i])^2+(\hat{y}[i]-y[i])^2} \quad (6)$$

where $(\hat{x},\hat{y})$ are the Cartesian coordinates of the center of the predicted bounding box, $(x,\ y)$ are the Cartesian coordinates of the center of the real position of the target and N is the total number of frames of the video.

4.2 Experiment 1: synthetic dataset

This first experiment consists on using the complete synthetic dataset to train the SiamFC and ensure that the implementation works well when detecting a simple type of target. All training parameters are fixed as indicated in Table 6.

4.2.1 Training

We observe that since the very first epoch of the training process, the loss becomes a NaN value. NaN stands for "not a number", and is a numeric data type representing an undefined or unrepresentable value that appears especially in floating-points operations. An extract of the training output showing this issue can be found in Fig. 9.

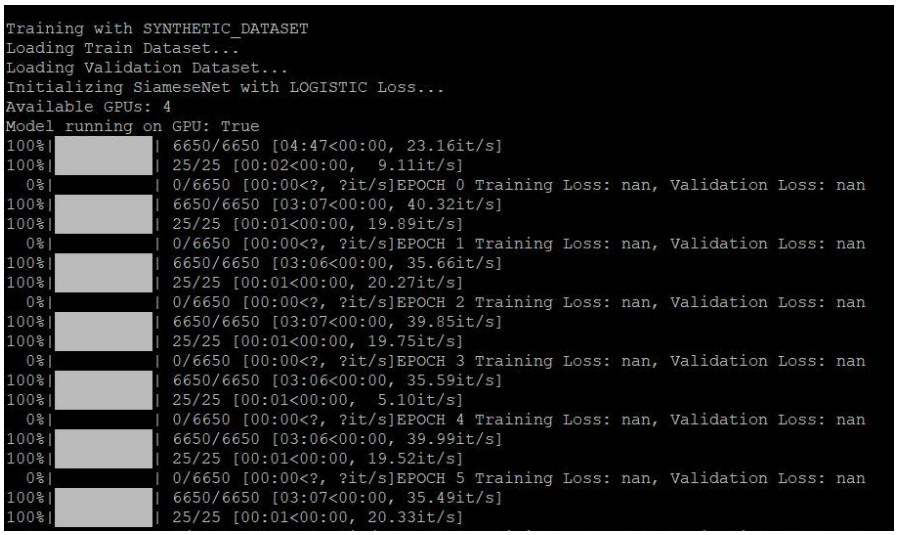

Fig. 9: Output of the training process showing the issue with NaN values.

Although we were not able to experimentally prove the cause of this problem, we strongly believe that is caused because the input of the logistic loss function, i.e. the values of the score map, is outside of the function domain. This can be caused either by a large negative number in a positive pair (i.e. when the ground truth is +1) or by a large positive number in a negative pair (i.e. when the ground truth is -1). In both cases, evaluating the function results in a $\log(\infty)$, which gives a NaN value.

Another potential cause could be a learning rate too high, which can cause the training to not converge or even diverge. However, we have empirically observed that decreasing the learning rate does not fix the problem.

The use of the logistic loss during training has also another issue. The function implementation in *Pytorch* does not accept any weights as input, which means that the weights defined in equation (5) cannot be applied to eliminate class imbalance. To use this

loss function with weights, a custom implementation should be developed, which is out of scope for this study.

The solution adopted for these issues is to replace the logistic loss function with a binary cross entropy (BCE) loss

$$L(y, v) = \frac{1}{|\mathrm{D}|} \sum_{i \in \mathrm{D}} \mathrm{y[i]} \cdot \log x[i] + (1 - y[i]) \cdot \log(1 - x[i]) \tag{7}$$

where $y[i] \in \{0,1\}$ is the ground truth label. Note that these values are different from the ones used with the logistic loss $(-1, +1)$, so the implementation of the ground truth mask must be also modified. The BCE loss is a suitable loss for binary classification tasks and is broadly used with siamese networks. We also noticed that most of the existing Python implementations of SiamFC [30], [32], [33] make use of this loss instead of the logistic loss.

The results show that the BCE loss is more stable during training and fixes the problem with NaN. However, we have not focused on mathematically demonstrating this change of behavior between losses. As shown in Fig. 10, the BCE loss has also inputs that are out of the function domain and could cause problems with NaN values.

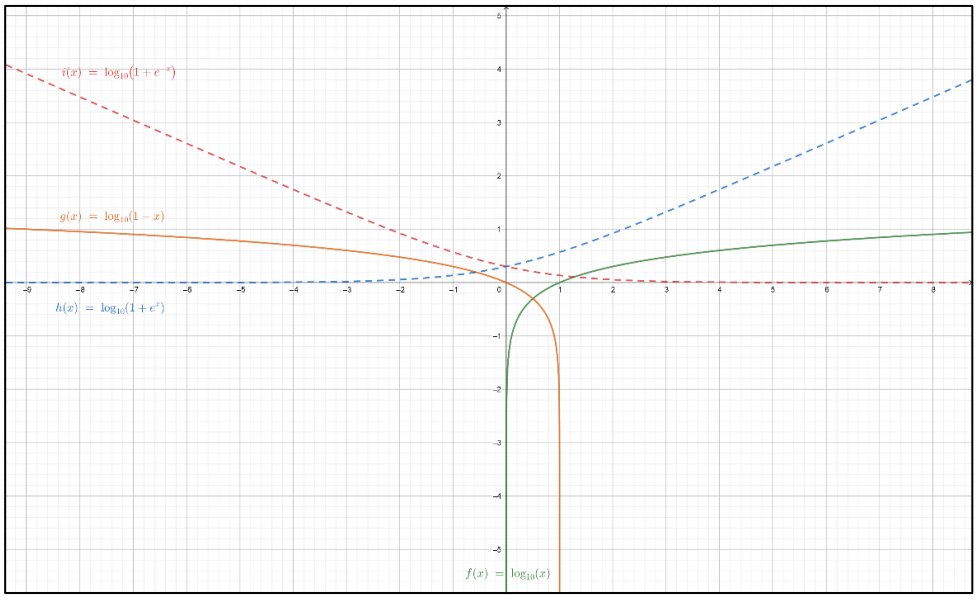

Fig. 10: Comparison between BCE and logistic loss. Red: logistic loss for $y[i] = -1$; blue: logistic loss for $y[i] = +1$; yellow: BCE loss for $y[i] = 0$; green: BCE loss for $y[i] = +1$

We can also observe in Fig. 11 that the validation loss shows a point of inflexion in which it starts to increase. This is a clear sign that the model is overfitting. The best validation loss is achieved in epoch 7. Since the implementation of the training process allows us to save the state of the model at each epoch, we use the state of the model at epoch 7 to perform the evaluation.

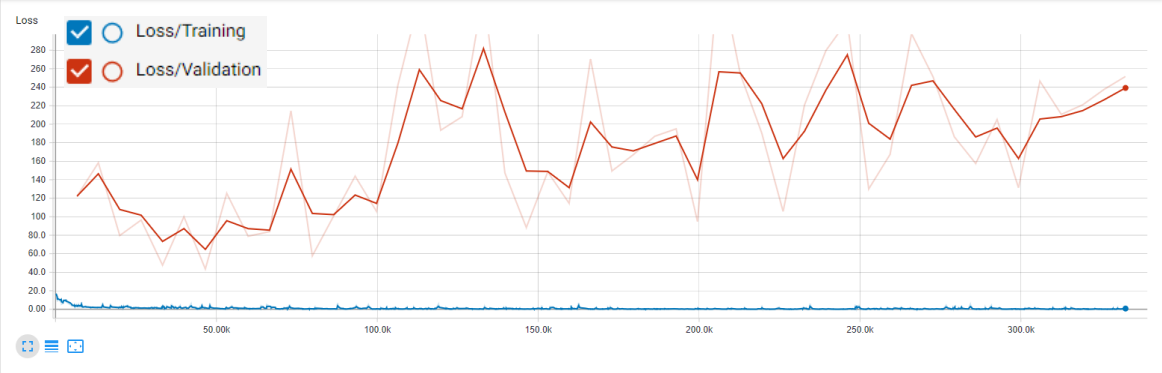

Fig. 11: Training and validation loss.

4.2.2 Evaluation

The performance of the tracker for this experiment has been evaluated using a synthetic video of 300 frames of size 255 x 255 pixels and a target of size 40 x 40 pixels. The maximum pixel displacement of the target between frames has been limited to 4 pixels. Since the size of the target is fixed for all the frame, we only search over 1 scale.

Table 7 shows the results of the evaluation. The overall performance of the tracker is very good, with a Center Error of 1.7 pixels.

	Center error (pixels)
Experiment 1	**1.70**

Table 7: Center Error for experiment 1

For this experiment, we also compute a graph showing how many frames have a certain number of pixel displacement in both coordinates (Fig. 12). The vertical displacement follows a normal distribution and there is no apparent bias. On the other hand, it seems that there is a little bias on the horizontal tracking that causes the tracker to put the bounding box slightly displaced to the left of the target. Despite this bias, the predicted bounding boxes fit almost perfectly the target in all the video frames (Fig. 13).

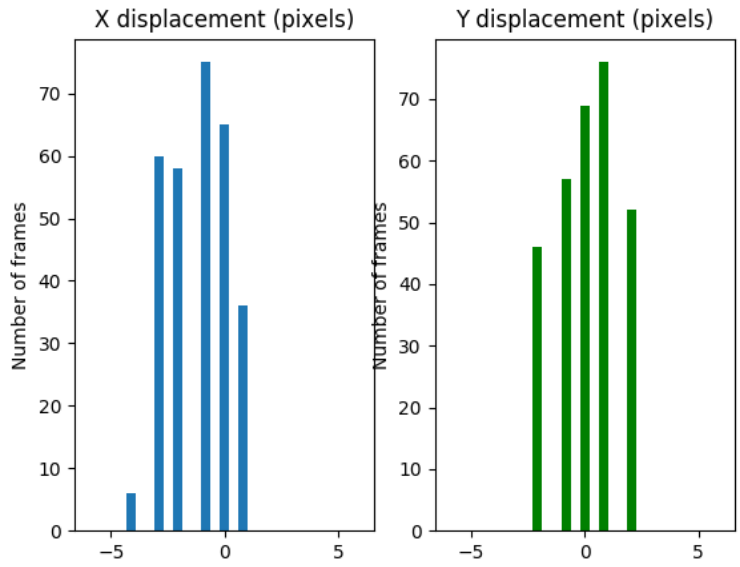

Fig. 12: Pixel error distribution for x and y coordinates.

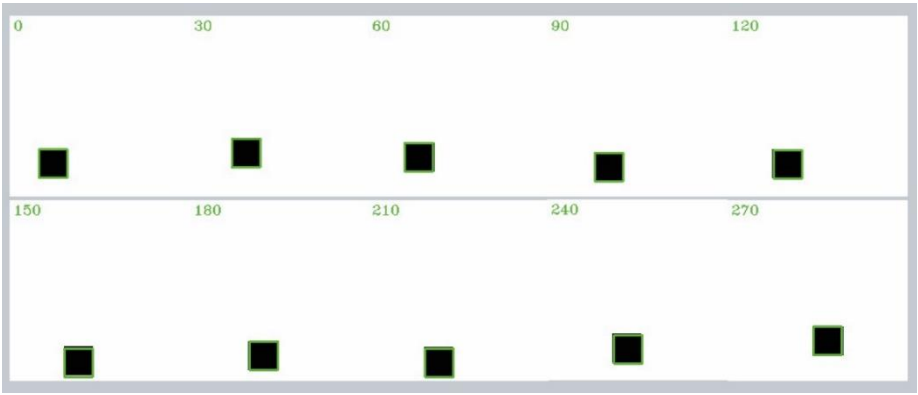

Fig. 13: Predicted bounding boxes (in green).

4.3 Experiment 2: ImageNet dataset

This experiment consists on training the network with all the 4417 videos of the ImageNet dataset for object tracking in video. The values for the training parameters are the same as in Experiment 1.

4.3.1 Training

As shown in Fig. 14, the validation loss does converge correctly when we train the network using ImageNet dataset. Although the training loss is higher than in the previous experiment, it is also closer to the value of the validation.

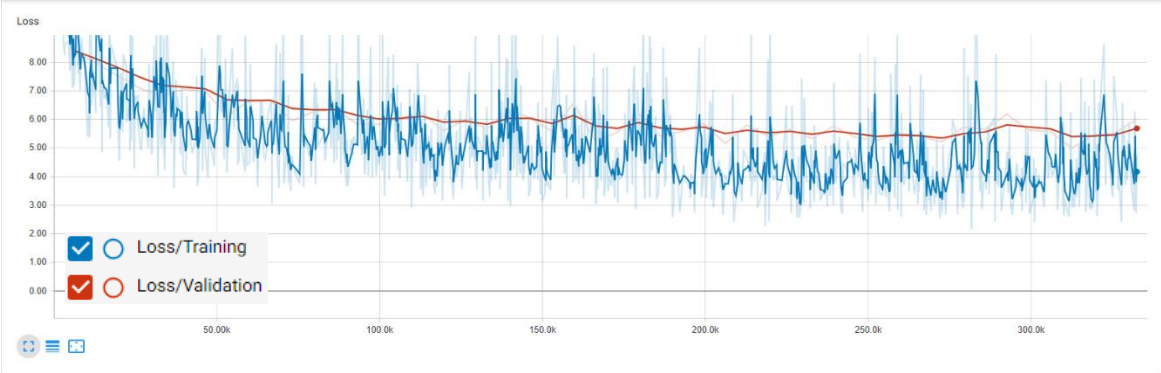

Fig. 14: Training and validation loss.

4.3.2 Evaluation

We evaluate the performance of the ImageNet-trained tracker by using the same synthetic video as in experiment 1. Table 8 shows an increase in the Center Error but still an overall good performance.

	Center error (pixels)
Experiment 2	**4.19**

Table 8: Center Error for experiment 2.

It is important to remark that the synthetic video has a rather difference appearance if we compared it to the videos that conform ImageNet. Observing the results of this experiment, the tracker seems to be able to generalise well and to track unseen objects that have no similarity with those used for training.

4.4 Experiment 3: tracker comparison

The goal of this third experiment is to compare the performance of both trackers from the previous experiments, i.e. the one trained with the synthetic dataset and the one trained with ImageNet, with different videos. In particular, we select 3 synthetic videos, 3 videos extracted from ImageNet and 3 videos extracted from LaSOT dataset. Table 9 shows a more detailed description of these videos.

Video	Video Source	Number of frames	Dimensions (pixels)	Description	Appearance changes	Motion
Video 1	Synthetic video	300	255 x 255	Black square moving on white background	None	Low
Video 2	Synthetic video	300	255 x 255	Black square moving on white background	None	Medium
Video 3	Synthetic video	300	255 x 255	Black square moving on white background	None	High
Video 4	ImageNet	464	1280 x 720	Still turtle	Low	Low
Video 5	ImageNet	264	1280 x 720	Flying helicopter	Medium	Medium
Video 6	ImageNet	419	1280 x 720	Dog running in the countryside	High	High
Video 7	LaSOT	1335	1280 x 720	Polar bear walking	Low	Low
Video 8	LaSOT	3000	1280 x 720	Boat in the sea	Medium	Medium
Video 9	LaSOT	2660	1280 x 720	Flying helicopter	Very high	High

Table 9: Details of selected videos.

The videos have been selected to include:

- A variety of object categories with different appearances.
- Different levels of appearance changes of the target during the video, such as rotations, scale and illumination variations.
- Different levels of target motion or camera movement.

Therefore, this experiment has two main purposes. First, we want to test how well the trackers generalise to new unseen objects. And second, we want to evaluate how the tracker deal with variations in the appearances and speed of the targets. Note that the three synthetic videos differ in the maximum displacement of the target from frame to frame. Video 1 is limited to 4 pixel displacement, video 2 to 16 pixels and video 3 to 32 pixels.

4.4.1 Evaluation

Table 10 shows the result of evaluating both models using the 9 selected videos for this experiment. The first thing we notice is that the tracker trained on the synthetic dataset performs surprisingly well on some of the videos from ImageNet and LaSOT with low or medium appearance variations.

This is rather an unexpected result as the synthetic dataset is relatively small and contains only one object category. In fact, the validation loss obtained during the training in

experiment 1 (Fig. 11) already indicates that the tracker did not learned to generalise to new unseen data. Despite this, the tracker seems to be able to perform tracking on other object categories under certain limitations. For example, in video 4 (ImageNet) and videos 7 and 8 (LaSOT) the tracker achieves similar performance to the ImageNet-trained tracker.

It is also clearly proved that appearance variations and fast motion of the targets affect the accuracy in tracking. To take a more in-depth look at the results, we include in Fig. 15, Fig. 16 and Fig. 17 some snapshots of the videos and the corresponding bounding boxes predicted by both trackers.

		Synthetic dataset tracker	ImageNet tracker
Video	**Video source**	**Center error (pixels)**	**Center error (pixels)**
Video 1	Synthetic video	3.95	4.19
Video 2	Synthetic video	4.18	17.70
Video 3	Synthetic video	4.34	18.72
Video 4	ImageNet	9.40	8.63
Video 5	ImageNet	253.90	41.45
Video 6	ImageNet	291.33	246.37
Video 7	LaSOT	14.38	9.54
Video 8	LaSOT	48.90	39.75
Video 9	LaSOT	388.94	302.06

Table 10: Center error for Experiment 3

We can see from the snapshots of the synthetic videos in Fig. 15 that the performance of the tracker trained with the synthetic dataset is excellent on the three videos. The tracker trained with ImageNet performs very well in the low motion video, but has some notorious bias on videos 2 and 3. It is important to keep in mind that maximum displacement of the target in video 2 and 3 is 16 and 32 pixels respectively. This is an unrealistic abrupt motion and we can expect some error in the results because the tracker applies a cosine window to penalize large displacements, assuming that the movement on a real object should be smooth and with no sudden changes. However, it seems that the error is due to a clear bias to the left on the horizontal prediction, similar to what we observed when evaluating the tracker trained with the synthetic dataset in experiment 1 (see Fig. 12). This bias seems to be a recurrent problem regardless of the dataset used during training.

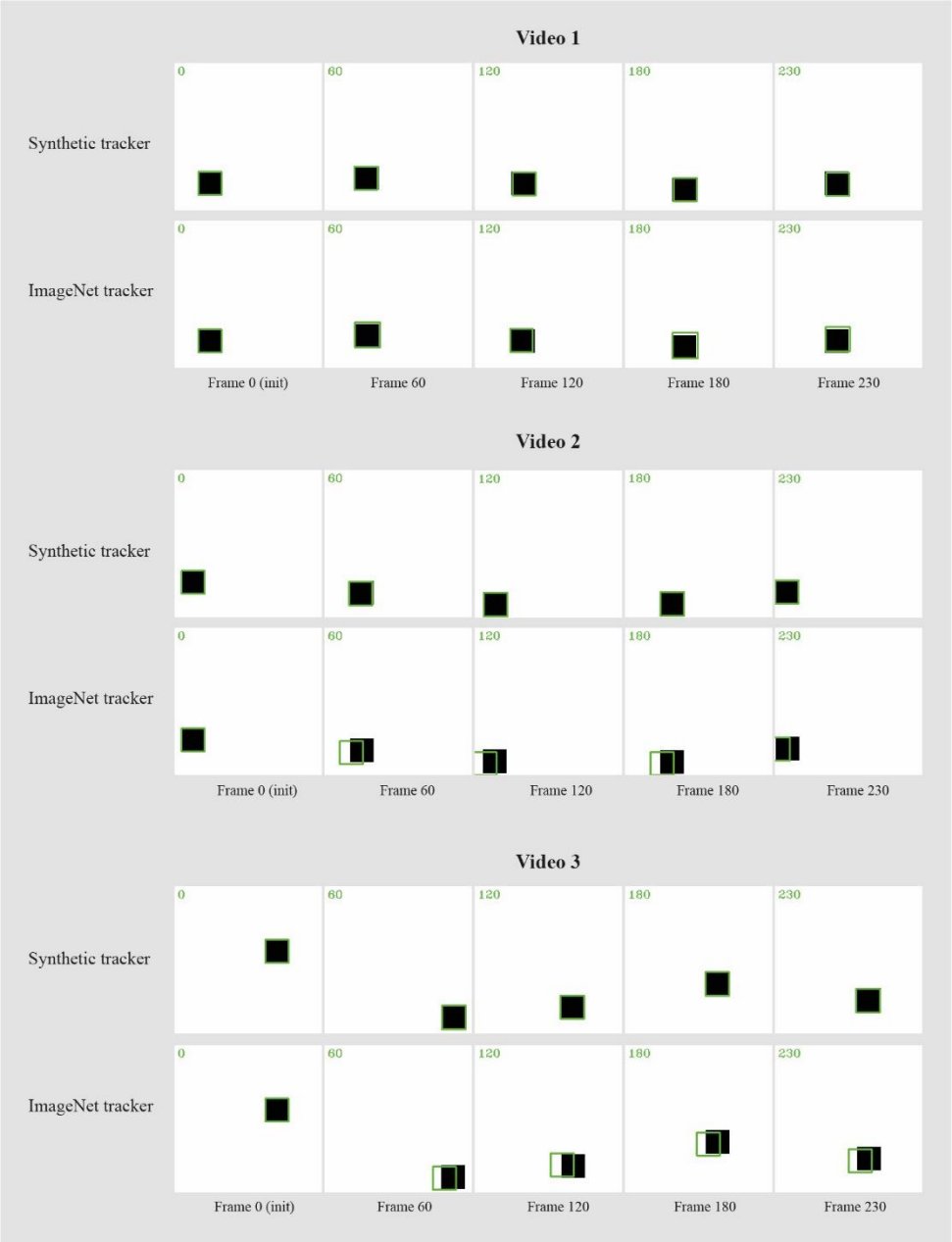

Fig. 15: Snapshots of the tracking result in synthetic videos

By evaluating the trackers on ImageNet videos (Fig. 16) we observe that both trackers perform very well in video 4, where the target appearance is practically the same throughout the video and the motion is low. While the tracker trained from synthetic data performs well only in this first video, the tracker trained on ImageNet also achieves some good results in video 5, which includes some rotations and variations in the direction of the target. Video 6 represents a complex case for object tracking where the target (the dog in frame 0) dissapears from the scene a second target from the same category (the dog in frame 60) enters the scene after some frames. If we observe the result of the ImageNet tracker, we can

see that it losses track of the first dog and it starts following the second dog after some frames. The task of differentiating targets of the same category and similar appearances in the same scene is quite complex, and in this case the tracker is not able to discern between the two targets. It is an expected result since this implementation of SiamFC is not desgined to deal with occlusions or multiple targets in the same video.

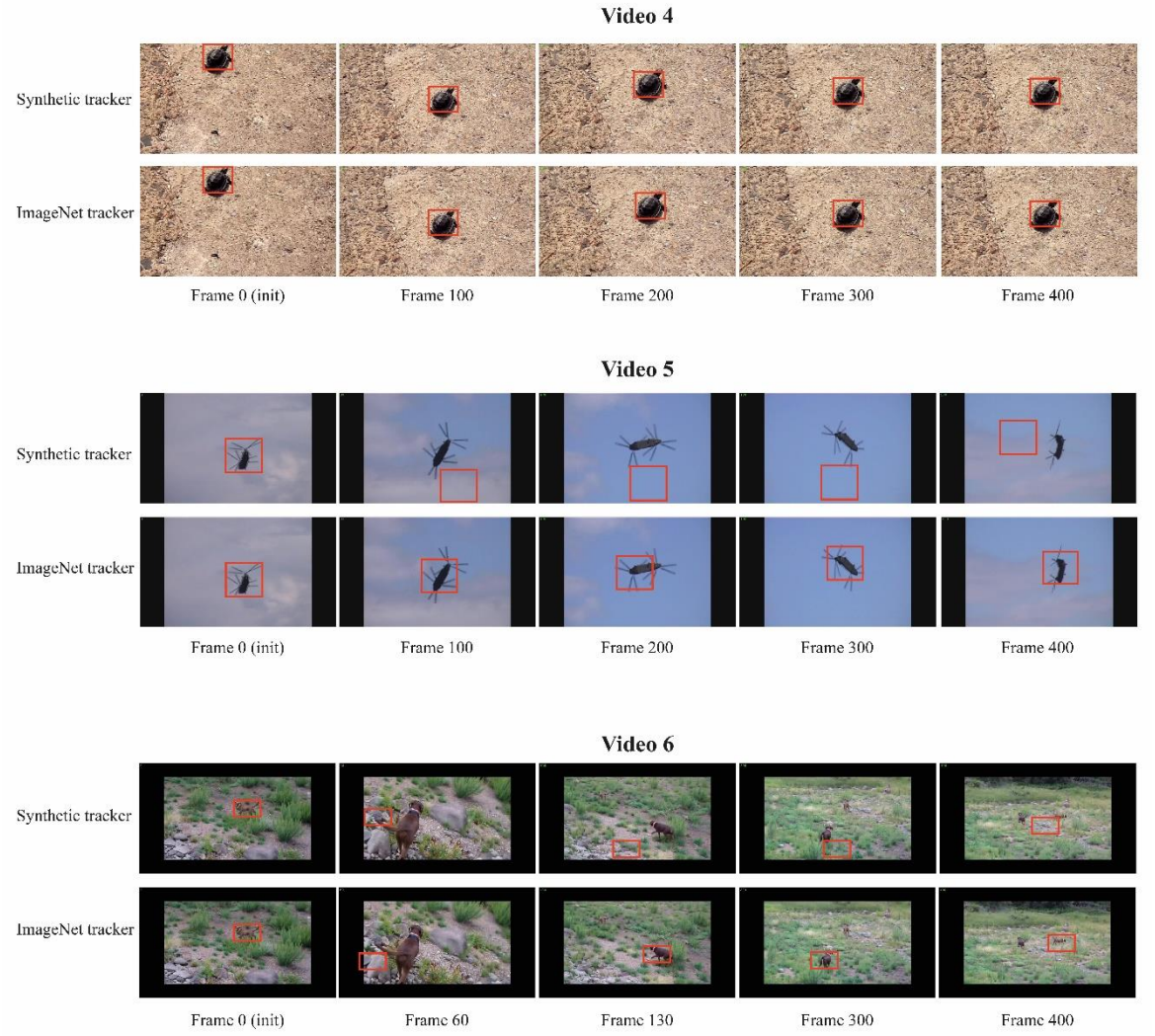

Fig. 16: Snapshots of the tracking result in ImageNet videos.

In the case of LaSOT (Fig. 17) both trackers achieve an overall good performance with low and medium appearance changes, i.e. video 7 and video 8. The last video represents a really complex case that includes not only changes in the target appearance, but also large-scale variations, changes in the background, illumination variations and fast motion. As a consequence, both trackers achieve poor accuracies.

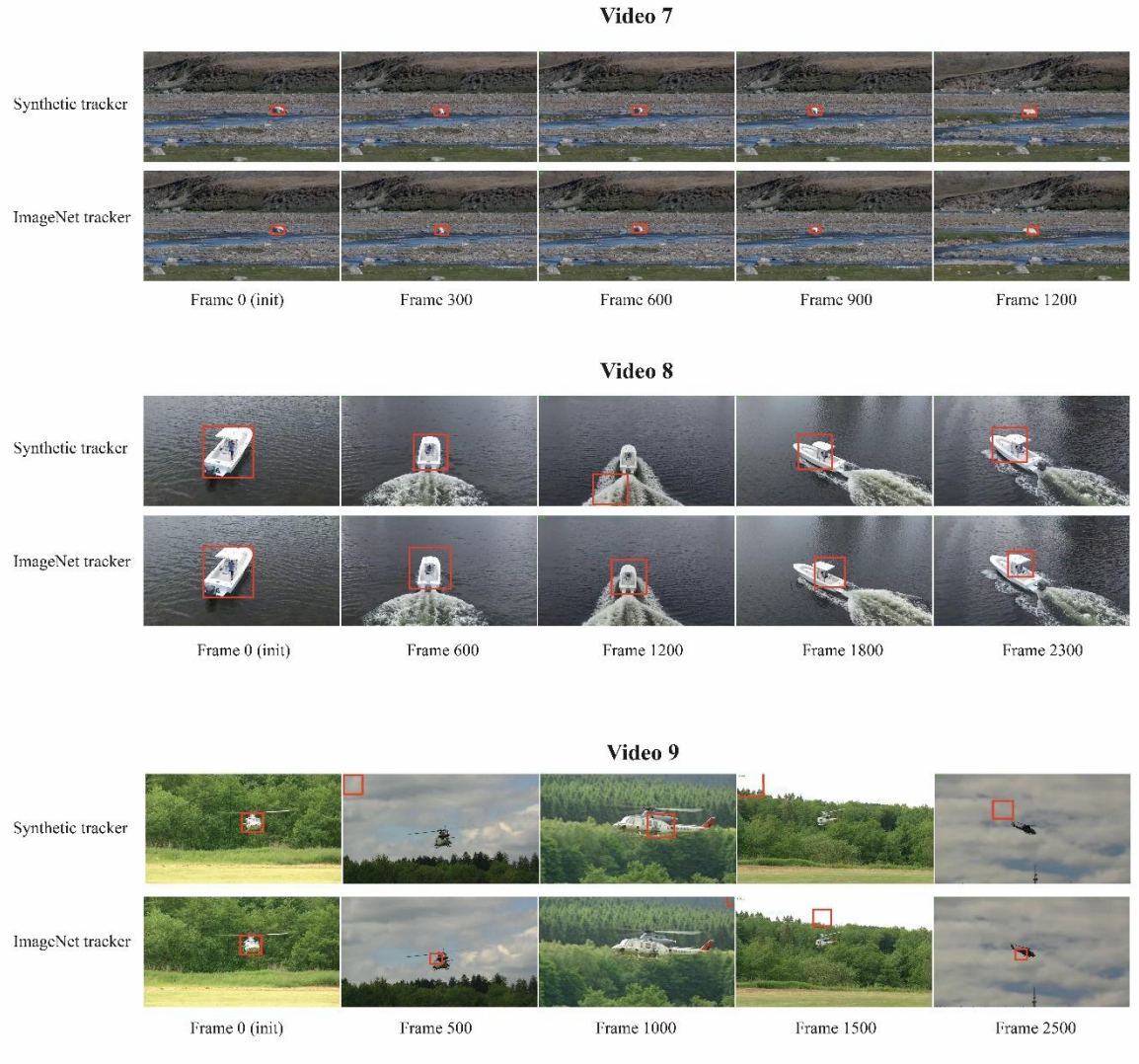

Fig. 17: Snapshots of the tracking result in LaSOT videos.

Budget

All the software used during the development of this thesis is open source and has not entailed any costs. The resources employed to carry out the experiments were provided by the Vienna Scientific Cluster for free. Table 11 includes an approximation of the costs of similar resources available on Google Cloud Platform[9].

GPU	Price (€/h)	Usage (h/week)	Weeks	Total
NVIDIA Tesla K80, 12GB GDDR5	0.12	30	28	**100.8 €**

Table 11: Cost for GPU usage on Google Cloud Platform

The main costs of this project come from the salary of the researchers involved. We consider my position as an undergraduate researcher and the supervisor of the thesis as a senior researcher.

	Amount	Wager/hour (€/h)	Dedication (h/week)	Weeks	Total
Undergraduate researcher	1	10	30	28	**8,400 €**
Senior researcher	1	35	2	28	**1,960 €**
				TOTAL	**10,360 €**

Table 12: Costs for researcher's salaries

[9] https://cloud.google.com/compute/pricing

Conclusions and future work

This work gives a general overview of the current tracking challenges and learning approaches and performs an in-depth analysis of SiamFC. SiamFC merges the power of Siamese networks and supervised learning to solve the problem of arbitrary object tracking.

Despite the simplicity of its architecture, SiamFC achieves state-of-the-art performance in multiple benchmarks. On the one hand, Siamese networks are the simplest networks for similarity problems and they are an excellent starting point to consider neural networks for object tracking purposes. These networks provide richer image features that affect directly the accuracy of the tracker. On the other hand, the use of deep learning techniques for tracking has grown since 2015 as it represents a new promising field of research. Deep learning has been proved to perform well in most of the problems it has been applied to. In the case of object tracking, it seems that this approach uses the available data more efficiently to solve the problem of tracking arbitrary objects in arbitrary scenes.

After performing the experiments, we conclude that SiamFC is really able to extend its tracking capabilities to a wide variety of object categories. We observe that the tracker trained with the ImageNet dataset has an overall better performance than the tracker trained with the synthetic dataset. This proves that SiamFC actually benefits from large datasets that contain different object categories. An interesting thing for further investigation could be to train SiamFC with some recently proposed datasets such as LaSOT or GOT-10K [16] and to see whether it benefits from their larger number of frames and object categories.

One of the big challenges that tracking has to deal with is the appearance changes of the target during the video due to rotations, scale changes, camera movement, illumination variations and noise. Experiments show that SiamFC is no exception and that the tracking performance under these conditions is sometimes poor. A potential improvement could be to take the target's appearance in the previous frame as the reference image instead of using the initial appearance for the whole video. This could have a negative impact on the speed of the tracker since it would need to compute the feature map of the reference image for every frame, but it also could help to better track targets with a high level of appearance variations.

SiamFC is a promising start of a new research topic in object tracking. The joint of deep convolutional networks and the increasing number of labelled data is, with no doubt, a major opportunity to finally solve the complex problem of arbitrary object tracking.

References

[1] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, “Safe driving envelopes for path tracking in autonomous vehicles,” *Control Eng. Pract.*, vol. 61, pp. 307–316, 2017.

[2] J. M. B. Oñate, D. J. M. Chipantasi, and N. D. R. V. Erazo, “Tracking objects using Artificial Neural Networks and wireless connection for robotics,” *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 1–3, pp. 161–164, 2017.

[3] and H. K. Y. F. Jiang, H. Shin, J. Ju, “Online pedestrian tracking with multi-stage re-identification,” *14th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, pp. 1–6.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “Fully-convolutional siamese networks for object tracking,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9914 LNCS, pp. 850–865, 2016.

[5] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A Survey,” *ACM Comput. Surv.*, vol. 38, no. 4, 2006.

[6] S. Ågren, “Object tracking methods and their areas of application : A meta-analysis,” 2017.

[7] M. Fiaz, A. Mahmood, and S. K. Jung, “Tracking Noisy Targets: A Review of Recent Object Tracking Approaches,” Feb. 2018.

[8] C. J. Veenman, M. J. T. Reinders, and E. Backer, “Resolving Motion Correspondence for Densely Moving Points,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, 2001.

[9] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, 2003.

[10] Jason Brownlee, “An Introduction to Feature Selection,” 2014. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/. [Accessed: 17-Apr-2019].

[11] S. Srinidhi, “What is Feature Selection and why do we need it in Machine Learning? – The Tech Check,” 2018. [Online]. Available: https://blog.contactsunny.com/data-science/what-is-feature-selection-and-why-do-we-need-it-in-machine-learning. [Accessed: 17-Apr-2019].

[12] D. P. Chau, F. Bremond, and M. Thonnat, “Object Tracking in Videos : Approaches and Issues,” *Int. Work. ”Rencontres UNS-UD”*.

[13] R. Pflugfelder, “An In-Depth Analysis of Visual Tracking with Siamese Neural Networks,” pp. 1–19, 2017.

[14] Max Pagels, “What is online machine learning? – Hands-On Advisors – Medium,” 2018. [Online]. Available: https://medium.com/value-stream-design/online-

machine-learning-515556ff72c5. [Accessed: 04-May-2019].

[15] Nikki Castle, "What is Semi-Supervised Learning?," 2018. [Online]. Available: https://www.datascience.com/blog/what-is-semi-supervised-learning. [Accessed: 11-May-2019].

[16] L. Huang, X. Zhao, and K. Huang, "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild," Oct. 2018.

[17] H. Fan *et al.*, "LaSOT: A High-quality Benchmark for Large-scale Single Object Tracking," 2018.

[18] R. S. Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, "Signature Verification using a 'Siamese' Time Delay Neural Network," *Proc. 6th Int. Conf. Neural Inf. Process. Syst. San Fr.*, 1994.

[19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verificatio," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1701–1708, 2014.

[20] P. Mukherjee, B. Lall, and S. Lattupally, "Object cosegmentation using deep Siamese network," Mar. 2018.

[21] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition," *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015.

[22] A. Das, H. Yenala, M. Chinnakotla, and M. Shrivastava, "Together we stand: Siamese Networks for Similar Question Retrieval," pp. 378–387, 2016.

[23] "Siamese Neural Networks." [Online]. Available: https://computervision.tecnalia.com/en/2018/07/siamese-neural-networks/. [Accessed: 20-Apr-2019].

[24] M. Kristan *et al.*, "The Visual Object Tracking VOT2017 challenge results," *VOT Work. 2017*, 2017.

[25] H. Fan *et al.*, "LaSOT: A High-quality Benchmark for Large-scale Single Object Tracking," Sep. 2018.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," pp. 1097–1105, 2012.

[27] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[28] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Sep. 2015.

[29] M. Nielsen, "Neural Networks and Deep Learning," in *Neural Networks and Deep Learning*, Determination Press, 2018.

[30] StrangerZhang, "SiamFC PyTorch," 2018. [Online]. Available: https://github.com/StrangerZhang/SiamFC-PyTorch.

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proc. Int. Conf. Artif. Intell. Stat.*, vol. 9, pp. 249–256, 2010.

[32] Rafael Eller, "GitHub - rafellerc/Pytorch-SiamFC: Pytorch implementation of

‘Fully-Convolutional Siamese Networks for Object Tracking,’” *2018*. [Online]. Available: https://github.com/rafellerc/Pytorch-SiamFC/. [Accessed: 05-May-2019].

[33] Huang Lianghua, “GitHub - huanglianghua/siamfc-pytorch: A clean PyTorch implementation of SiamFC,” *2018*. [Online]. Available: https://github.com/huanglianghua/siamfc-pytorch. [Accessed: 05-May-2019].

[34] S. J. Rao, Y. Wang, and G. W. Cottrell, “A Deep Siamese Neural Network Learns the Human-Perceived Similarity Structure of Facial Expressions Without Explicit Categories,” *CogSci*, 2016.