UNIVERSITY OF UDINE
hic sunt futura

Department Polytechnic of Engineering and Architecture

PH.D. THESIS IN
INDUSTRIAL AND INFORMATION ENGINEERING

Visual Object Tracking with Deep Learning

CANDIDATE
Matteo Dunnhofer

SUPERVISOR
Prof. Christian Micheloni

Cycle XXXIV — A.Y. 2020-2021

Institute Contacts
Dipartimento Politecnico di Ingegneria e Architettura
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
`https://dpia.uniud.it/`

Author's Contacts
Matteo Dunnhofer
Via delle Scienze 206
33100 Udine — Italia
`matteo.dunnhofer@uniud.it`
`https://matteo-dunnhofer.github.io`

A Elisa.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Christian Micheloni for his constant support in the research activity and all the opportunities he gave me. His guidance and vision will be always sources of inspiration for me. I would like to also give thanks to Prof. Matej Kristan and Prof. Emanuele Frontoni for their precious comments on this Thesis.

A big “thank you” must be devoted to the present and past members of the Machine Learning and Perception Lab for the nice times spent together and the always fruitful discussions: Niki, Matteo, Rita, Asad, Rao, Pankaj, Vaibhav, Tok-Tok. I would like to thank also Matteo Miani for the productive moments spent working together.

A very big thanks goes to my parents, Giovanna e Maurizio, and to my whole family for the neverending support and encouragement that began well before this chapter of my life. They set the basis for this dream to become true and for the person I became today. Thank you.

I have to be really grateful to my beloved Elisa. She constantly encouraged and supported me in achieving my dreams and ambitions, and she helped me through every day things unconditionally. The merit of this work is in part hers and I hope it will pay off her efforts. Finally, I have to say a little thank you to a little cat called Merlino which continuously kept me smiling while working at home.

Abstract

In its simplest definition, the problem of visual object tracking consists in making a computer recognize and localize persistently a target object in a video. This is a core problem in the field of computer vision that aims to replicate the human ability in keeping the focus on a particular object with the sight. In the past, several different algorithmic principles have been proposed to reach such a capability. Thanks to the tremendous improvement in accuracy, recent algorithms based on deep learning emerged as promising methodologies to achieve the goal. The fundamental idea behind these techniques is to exploit the ability of deep neural networks in learning complex functions to learn how to track objects by visual examples. The potential of this kind of tool attracted the interest of the research community so much that nowadays deep learning is the way-to-go for the implementation of effective visual tracking algorithms. Despite the popularity however, the study of deep neural networks for visual tracking is relatively at its early stages. This means that there are still many open issues that need to be addressed to fully comprehend the capabilities and potentialities of such learning-based models. In this Thesis, we try to give answer to some of these questions. The manuscript will start by dealing with the problem of the inefficiency of the two-stage procedure employed by visual tracking methodologies that used reinforcement learning to optimize their deep neural networks. The contribution that will be presented is based on concepts of imitation learning and substitutes the learning procedure with a single end-to-end learning strategy, making the learning of a tracking policy easier and more effective. Such an idea of learning tracking from another tracker will be later generalized in a new deep learning-based framework that explicitly considers multiple trackers as sources of information. This framework, which marries notions of knowledge distillation and reinforcement learning, aims to unify application objectives such as fast processing speed, accurate online adaptation, and fusion of trackers, that were reasoned independently before. After the discussion of the importance of building upon the knowledge of multiple and complementary trackers, we will exploit such an idea to tackle long-term visual tracking, a more challenging setting in which the objects to be tracked are allowed to disappear and re-appear in the video without constraints. An effective award-winning deep learning methodology will be introduced to fuse the characteristics of two complementary state-of-the-art trackers in such a scenario. The Thesis will then move on by focusing on domain shift and overfitting, two critical issues affecting deep learning models that were not studied much in the context of deep learning-based tracking applications. A weakly-supervised domain adaptation strategy based on the tracking-by-trackers framework introduced before will be presented to address the aforementioned difficulties. Reinforcement learning will be used to express weak supervision as a scalar application-dependent and temporally-delayed feedback, and knowledge distillation will be employed to guarantee learning stability and to compress and transfer knowledge from more powerful but slower trackers. The next study will concentrate on a domain which resulted particularly challenging for deep learning-based and non visual trackers: First Person Vision (FPV). This is a sub-field of computer vision devoted to the study of algorithms processing images and videos acquired from cameras mounted on the head of a person. Such a setup permits to replicate humans' first person viewpoint in a computer. Despite a few previous attempts to exploit visual tracking in this domain, a methodical analysis of the performance of state-of-the-art tracking algorithms was missing. We will hence present the first systematic study of visual object tracking in FPV. The investigation extensively analyses the performance of different methodologies and suggests that more research efforts are needed for this problem so that tracking could benefit FPV applications. But all the studies that will be presented until this point assume that bounding-boxes – rectangular shapes expressing just the position and scale of the targets – are employed to represent the state of the target objects in the videos. Towards the end of the Thesis, we will try to move away from such a representation. We will first present a study over deep learning-based segmentation methods that can be conditioned on the target object to transform any bounding-box tracker into a segmentation-mask tracker, i.e. a tracking algorithm able to provide the precise position and shape of the object at the pixel-level. Secondly, the problem of knee cartilage tracking during ultrasound-guided minimally invasive procedures will be investigated. This is a particular tracking problem in which precise information regarding targets in the form of segmentation masks is required, and a new deep learning methodology will be presented to address it. Extensive performance validation will demonstrate that our proposed algorithm is able to track the cartilage with an accuracy comparable to experienced surgeons.

Contents

1 Introduction

The *sight* is the sense humans mostly rely on to perceive the world [1]. Indeed, through the stimuli received by the eyes our brain is enabled to understand the world we live in. This is achieved by the ability of such an organ in processing the input stimuli to obtain object recognition, object localization, and motion sensing. These are all fundamental unconscious processes in the formation of more complex cognitive capabilities that ultimately lead to the emergence of high-level behavior such as driving a car or play tennis.

Towards the implementation of artificial intelligence systems able to act as autonomously as humans, machines need the ability to perceive the world as we do. And given the importance of visual perception for us, we can easily imagine that the sight must be somehow incorporated into such artifacts. The quest to how to solve such a non-trivial problem dates back to the 1966[1] and it gained increasing interest around it during the years so that nowadays it forms the basis of a scientific field known as *computer vision*. The goal of this community is to study and develop computerized methods to make machines sense and understand the world from pictures of it. Making a computer see is a problem that requires the resolution of many different technological challenges. Indeed, to effectively understand the dynamics of the world a computer needs to first capture the state of the world with some kind of visual sensor, for example a video camera. Such information must be then represented digitally and stored in the computer memory. Only after, the information can be processed to get high-level knowledge about the world. In this pipeline, which includes problems related to other scientific disciplines such as electronic engineering and information theory, the latter aspect is what computer vision algorithms try to address. But it turns out that directly automatizing high-level behaviors, such as driving a car, from the raw and high-dimensional data acquired by video cameras is impracticable. To successfully solve real-world problems, the vision system of a computer should be organized in a pyramid of processes aimed to understand increasingly abstract characteristics of the world. For example, the recognition and localization of colors or shape patterns in images can be

[1]`https://people.csail.mit.edu/brooks/idocs/AIM-100.pdf`

considered as low-level tasks whose outcome can be exploited for the recognition of more abstract entities. Indeed, knowing which particular shapes are visible and the relation between them allows to retain higher-level information such as the presence and position of an object. Let's make it more concretely. If we take into consideration a Formula 1 car, we notice that we can recognize it by the particular shape of its body, by the round shape of the wheels, and by the color patterns of the Formula 1 team's look style or of its sponsor banners. If we then consider that the world is dynamic and not static, we realize that much more information is added due to the motion and the dynamics of moving patterns. This allows not only to recognize an object but also to understand its movements in relation to the surrounding environment. This information can be ultimately used to generate even higher level knowledge about a Formula 1 car, for example to understand that it is approaching a curve or that is surpassing another car.

As it can be inferred from this description, there are many problems to solve in order to make a computer vision system work. Given the complexity of the real world, each problem requires dedicated study to make the overall system effective. This Thesis focuses on the task that requires the recognition and localization of an object of interest in a video. In computer vision terms, this is referred to as the *visual object tracking* problem.

1.1 Visual Object Tracking

Following an object with the sight its one of the ability an human leverages on every day. It is an almost unconscious task that is the basis of more complex cognitive processes performed by our brain [2]. Indeed, keeping track of objects and things in terms of their appearance and motion allows us to understand the world's dynamics and behave accordingly. Considering the particular information it can deliver about objects, such an ability is desired for computer vision systems to have. Indeed, many practical implementations of intelligent systems require the automatic following of an object of interest. For example, video surveillance systems use such kind of algorithms to understand the motion of people in a monitored area for security reasons. In the sports analytics domain, visual tracking programs are employed to calculate how precise the motion of an athlete's body is in the execution of exercises. In medicine, surgical robots leverage specific algorithms applied to medical images to continuously know the positions of human internal structures in order to perform safe and efficient operations.

In this section, the particular characteristics of the visual tracking problem – also referred as visual tracking – in the context of computer vision will be presented.

1.1.1 Problem Definition

There is no formal definition of the problem of visual tracking on which the computer vision community completely agrees on and it has been conceived by different terms that however share common concepts. Yilmaz et al. [3] stated that "Tracking is the task of assigning consistent labels to the tracked objects in different frames of a video". Cucchiara et al. [4] provided the following definition "Online tracking is following the location of one target in a video starting from a selected region of interest in the first frame" that has been later generalized as "the task of giving spatial and temporal

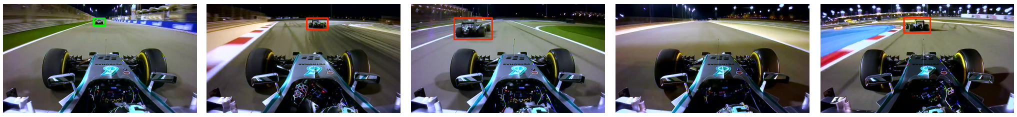


Figure 1.1: Example of a visual object tracking problem. The images are sampled from the frames that compose the video acquired from the camera mounted on a Formula 1 car. The target object to be tracked by an algorithm is the other Formula 1 car in front of the chasing one. The algorithm is first given the target state (represented in green in the leftmost image). Then, it should be able to recognize the object of interest and provide its localization by proposing its successive states (represented by the red rectangles) for all the frames. The object could also disappear from the scene captured in the video and hence not being present in some frames.

coherency to the object identification among the time and in the space". Maggio and Cavallaro [5] defined the problem of tracking a single object in a video as the estimation of a time series representing the states of a target object in the consecutive frames.

There are two major points to take away from these descriptions: it is assumed that the initial state of the target is given, in other words, a target object is initially selected with some kind of mathematical structure representing its position and/or shape; solving the problem means delivering a consistent referral to the object in terms of the given structure and that should be maintained for all the video long. Figure 1.1 shows an actual example of a visual object tracking task. In this case, the target is identified in the first frame of the video (the leftmost image) with a rectangle structure (highlighted in green) enclosing the pixels belonging to the Formula 1 car in the front of the chasing one. Visually tracking such a car means providing the rectangles enclosing its pixels (red rectangles) while the target car's appearance and position change.

Hence, to achieve its goal, a visual object tracking algorithm – referred also as a visual tracker or tracker – must analyze every single image that composes a video (i.e. the frames), extract relevant information about the target object and the relative scene, and produce an output – termed state – that carries spatially and temporally consistent information about the object's motion.

1.1.2 State Representation

We already mentioned what is an object from the perspective of computer vision. Its state is a representation that summarizes relevant semantic information about the object. For example, the horizontal and vertical coordinates of a point located at the barycenter of the object provide information to determine its position in a single frame. Considering the variations of the coordinate values across multiple frames, we can obtain information about the motion of the target. A type of representation providing richer information is the bounding-box, that, in addition to the coordinates to represent the target's position, includes width and height values. These entries are exploited to define a localized rectangle that encloses the image pixels belonging to the appearance of the target, ultimately providing not only the position but also an estimate of the scale of the target. Even richer representations are those referred to as segmentation masks. These are images with binary values that express which pixels of the input frame belong to the

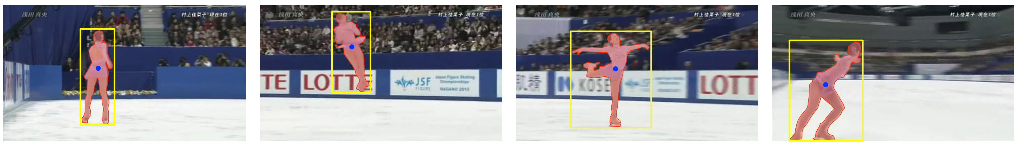


Figure 1.2: Visual example of the different model-free target state representations. In the frames sampled from this video the target object is the ice-dancer. The blue dots represent the barycenter of the target, the yellow rectangles the bounding-boxes, and the red figures the segmentation masks which highlight the image pixels belonging to the target.

target appearance and which do not. This kind of representation allows to determine very precisely the position and the shape of the object of interest. Figure 1.2 reports visually how the aforementioned representations vary when they are applied to the same target object. All the aforementioned representations are referred to as model-free. This is because they are based on mathematical models that are independent from the object's category and essence. By exploiting such a type of representation it is possible to model a multitude of different objects and consequently develop trackers able to manage and track arbitrary objects.

There exist more sophisticated target representations that are designed to give more detailed information about the motion of targets. For example, skeleton and N-DoF (Degree of Freedom) representations are employed to obtain precise information about the motion of objects of particular categories (e.g. skeletons are used for the tracking of the position and pose of people). In general, such state models are grounded on particular assumptions related to the appearance and motion of objects and hence are suited for the development of algorithms capable of tracking only particular types of objects.

1.1.3 Challenges

Several different challenges must be faced by an algorithm that aims to accurately keep track of the position of an object. Some of these are related to the scene captured by the camera. They include the variation in the illumination, the color changes, and the presence of similar objects. Other challenges are instead related to the behavior of the target. Examples of such are pose and scale changes, rotations, shape variations, and fast movements.

But the hardest events to address are definitively the occlusions of objects. These are situations in which the target object is partially or sometimes even totally hidden by another object, or it is partially or completely out by the camera's field of view. The fourth frame of Figure 1.1 shows an example of the latter situation. The difficulty to overcome these challenging factors breaks the problem of visual object tracking into two categories that are referred to as short-term and long-term tracking respectively [6]. A short-term tracking problem is built upon the assumption that the target never disappears completely in any frame of a video. This means that it is never totally occluded by another object or it never leaves the field of view of the camera. Such a setting limits the impact of occlusions and allows an algorithm to focus more on addressing the other challenging factors. On the other hand, when the assumption of continuous target presence in the frames is dropped we talk about long-term visual tracking. In this scenario the target is allowed to disappear and re-appear. This is certainly a more challenging setting that requires a tracker not only to overcome the other challenges but also to provide mechanisms to detect the disappearance and to find again the target when it reappears in the scene.

In addition to the aforementioned challenges, constraints regarding the computational efficiency are posed to tracking algorithms by many applications. Just like the visual tracking process performed by humans that is an underlying task for more complex cognitive tasks we perform in real-time, also the computerized counterpart needs to be performed as fast as possible. Indeed, the output produced by trackers is often needed by other algorithms that must produce a higher-level semantic output based on the motion of the target. Hence, beside being accurate, visual tracking algorithms should be efficient in the usage of their resources and in the implementation of their routines.

1.2 Deep Learning

Another fundamental ability of humans is *learning*. Since our birth, we are constantly surrounded by the world environment which we perceive through our senses. The sensory information is constantly stored and processed by complex cognitive procedures such that higher-level knowledge is retained. Then we are able to adapt the latter to every new context in which we live in. There is no doubt that this is one of the most important abilities of our specie, since it allows us to acquire new skills and discover new things, ultimately making us behave more and more intelligently. We can say that learning is a manifestation of intelligence. Considering the importance of such an ability, it is easy to understand why the research community in artificial intelligence spent a surge of efforts for decades in order to replicate such an ability inside a computer. Despite the great progress made since the dawn of computer science and artificial intelligence, we are still very far to replicate the knowledge acquisition process from experience inside an artificial system as humans do. The complexity of such an ambitious mission does not mean that computerized learning strategies are not possible at all. Indeed, algorithms to make computers learn have been used for decades to solve complex but narrow problems. The fundamental idea behind these solutions is to develop algorithms capable of automatically extracting knowledge from raw patterns of sensory data representing some characteristics of the problem and of taking decisions based on abstract representations of them. The area of research devoted to these practices is called *machine learning*.

1.2.1 Machine Learning Basics

There are a few fundamental concepts that form the basics of this scientific field. First of all, in every machine learning problem we have the dataset. This can be viewed as a group of entries containing structured information about the problem. Each entry, which is a sample, corresponds to a set of values – called features – which summarize key characteristics of an observation of the problem. The dataset is split into two subsets, the training set, and the test set. The former is generally larger and comprises all the samples that are devoted to learning, i.e. that are employed to extract knowledge from the information contained in the features. To achieve that, different mathematical strategies exist depending on the amount of semantic information associated with the features. Such kind of information defines the objective of the learning task and also provides a categorization for it. Indeed, we talk about a supervised machine learning problem when each of the samples is associated with a ground-truth annotation that expresses the correct answer that should be presented when the sample's features are observed. In this annotation regime, the goal of a learning algorithm is to acquire knowledge on how to provide an output that replicates the ground-truth annotation when inputted with the respective features. This setting is said to be supervised because the ground-truth answer is a rich representation of the desired output of the learning system. However, in many applications, it is really expensive to obtain rich ground-truth information to feedback the learning algorithm. To tackle such an issue, a ground-truth with a weaker form, such as a scalar value representing whether the algorithm is behaving well or not, could be used. In this scenario, we deal with a reinforcement learning problem and the goal of the system is to learn how to take optimal decisions given the information contained in the features and the limited feedback associated. If no ground-truth or feedback is present at all, the problem is said to be of unsupervised learning. In such a scenario, the learning algorithm should analyze the information available in the relations between the different samples in order to detect common patterns that would lead to the generation of knowledge.

Let's give an example to make things all the described concepts more clear. Consider the problem in which you want to develop an automatic system to perform the university exams in your place. A machine learning solution would require the learning of an algorithm that when inputted with an exam returns the (possibly right) answers to the exercises of the exam. So, we build a training set of exams. Each exam would be a sample, and the features could be some keywords appearing in the questions. Depending on the amount of feedback we want to give to the learning system, we can tackle the problems by the different paradigms described before as follows. For a supervised learning setting, we would associate an exam with the correct solutions to all the questions. This would be the best scenario since our algorithm would have the opportunity to compare its proposed answers with the correct ones and use the difference between the two to correct its predictions. For a reinforcement learning strategy, we would associate each exam just with its grade. This would result in a more challenging setup since the algorithm should try to figure out by itself which of the answers was responsible of the final grade, and update its behavior accordingly. For the unsupervised case, we do not associate any kind of feedback with the exam. This is for sure the most difficult scenario. Indeed, the algorithm would need to implement strategies to understand the relations between similar questions in order to provide consistent and logical answers across all exams.

The test set of the dataset generally associates features to the ground-truth annotations and it is used to evaluate the performance of the algorithm after learning. This subset should be designed to represent well the application scenario in which the learned program will be deployed.

1.2.2 Neural Networks

Until now we talked about an algorithm able to learn something from data in very general terms. Let's add some details and discuss how it is possible to implement such an intelligent program. Several different strategies have been proposed in the past to adapt a system to a set of input features. Among the many that include logistic regression [7], decision trees [8], and support vector machines [9], artificial neural networks [10, 11] emerged as one of the most versatile methodologies. This kind of learning algorithm resembles the adaptation process performed by the networks of natural neurons present in the mammals' brain.

Particularly, each artificial neuron mimics in a very abstract manner the process of natural neuron activation based on the activity of connected neurons. In formal terms, an artificial neuron is a mathematical function that computes a weighted sum of a finite set of sensory inputs. Such an operation is performed to gather the excitations of nearby neurons. After that, a so-called activation function is employed to threshold the amount of firing that should be passed to other neurons connected with it. The mathematical formulation of this idea dates back to the forties and was published in the famous paper about the perceptron model [10].

To perform sufficiently well in learning real-world tasks, the artificial neurons are arranged in a network organized as a stack of layers. Each layer is composed of a set of parallel neurons. Each of these is connected to all the neurons present in the layer coming before. At the first layer, the feature values are considered as neurons and hence each neuron of such layer is connected to all the features. In this way, every neuron receives input from all the sensory data. The connections are considered as scalar values – the set of these values is said weights – and are initially set to random values. During the learning phase, such values are adjusted in order to adapt to the task of interest by exploiting its data representation. The final layer of the network comprises neurons that only receive signals and that do not propagate further. The value computed by them is considered to be the output of the neural network.

The objective of this learning architecture – referred to as multi-layer perceptron – is to discover the optimal weight values that lead to the maximization of the learning system's performance in the desired task. This goal is achieved algorithmically in two major steps which are referred to as forward and backward pass respectively. In the first phase, the input features are presented to the neural network. Each of the first layer's neurons computes their weighted combination, applies the activation function, and returns the resulting values as output. Such an output is used as input by the second layer which applies the same procedure. The whole process is repeated until at the last layer of the network the output is composed. The backward pass starts from this point and proceeds backward through the network. First, the activation values produced by the network as output are compared to the feedback information available – referred to as target value – via a so-called loss function. This operation computes some kind of difference between the predicted and target values. Such a difference is exploited to adjust and optimize the network's weights through a simple but effective technique known as backpropagation [12]. In short, this algorithm computes the derivative of the loss function with respect to all the network's weights thanks to the chain rule, which provides a formulation to obtain the derivative of all the operations involved in the network. The values obtained by this procedure are called gradients and they express the direction in which each of the weights should be changed in order to minimize the loss function. Such information can be exploited to adjust the capabilities of the network to achieve the behavior of interest represented by the feedback associated with the features. The overall learning of the network is obtained through the iterative application of described operations until the model's capabilities match the desired level of quality.

1.2.3 Deep Neural Networks

The human reasoning process is built on hierarchies of concepts in which more complex ones are obtained through the relation and combination of simpler ones. Such a cognitive ability is also exploited during learning. We first learn simpler things and then use them to learn new and more complex things. It turns out that, to some extent, such an effective scheme is replicated by artificial neural networks, and it explains in part why these learning systems are able to learn very complex tasks. Indeed, the recent usage of neural networks organized as stacks of many layers of neurons (up to thousands) achieved breakthrough results in many scientific fields including speech recognition [13], natural language processing [14], biology [15], computer game play [16], finance [17]. The effectiveness of this kind of method attracted the interest of researchers and practitioners so much that the study of neural networks became a major research field that today we call *deep learning*.

Among the field of application that pulled the deep learning revolution, there is computer vision. This happened thanks to the breakthrough achieved by AlexNet [18], a deep neural network architecture that has been used to win the ImageNet challenge [19]. This was a competition in which an algorithm had to discriminate images based on 1000 categories. In 2012, AlexNet won the challenge with a huge gap over the second-best solution. More importantly, even though the idea of using particular neural network architectures has been for since decades, such a result has been the first demonstration of deep neural networks' abilities in addressing very complex problems. Kryzhevsky et al. [18] proposed a neural network composed of so-called convolutional layers. These are special layers of neurons able to effectively retain knowledge from spatially organized data such as images. The underlying working mechanism was previously introduced by LeCun et al. in the nineties [20]. However, AlexNet has been the first solution proving that such kind of deep network was able to perform very well in real-world problems when trained with large amounts of data on dedicated computing platform such as GPUs.

Interpretation strategies [21] applied to neural networks like AlexNet later revealed that such kinds of networks resemble the working mechanism of the human visual system. Indeed, after the training on images, learned deep convolutional neural networks show to rely on hierarchies of visual features that represent increasingly abstract concepts. Such a characteristic enable these kinds of models to build high-level representations of complex objects in an automatic way starting from lower-level concepts such as edges, shapes, and colors.

The successful application of deep learning to the ImageNet challenge inspired the exploitation of deep neural networks so much that nowadays this is the go-to methodology for many different computer vision tasks such as object detection or semantic segmentation. As will discuss shortly, even the problem of visual object tracking benefited from this revolution.

1.3 Literature Survey

Several different principles have been exploited in the past to develop algorithms capable of tracking objects in videos. In this section, we review briefly the most relevant approaches that have been exploited until today.

1.3.1 Key Ingredients

Before surveying the solutions, we provide the reader with some notions about the main conceptual components that form a visual tracking algorithm. These can have different form in the implementations or be implicit in some methodologies, but they can be found in every tracker and they help in understanding its working mechanism.

The first important concept is the template. This term is used to refer to the visual information regarding the target and highlighted by the ground-truth state given in the first frame of the video. In practice, such information consists of the pixels distribution of the image patch containing the initial appearance of the target object. The template is used by the trackers to maintain a reference to the true target while they analyze the video.

Another important notion is the appearance model. This is used to refer to the mathematical model employed to obtain abstract information about the visual appearance of the target. Such a model is applied during the tracking phase to distinguish the target pixels from those of the background. Different strategies can be used to implement appearance models, e.g. color histograms, gradient-based features [22], the features produced by convolutional neural networks [20], or a combination of them. The feature extraction operations can be also used in conjunction with patch sampling strategies. These are used to select one or more sub-regions from the template image to build coarse-to-fine model representations. In general, the appearance model should be robust enough to provide information invariant to the target's scale and pose changes, illumination variations, or partial occlusions.

The motion model comprises all those assumptions regarding how targets are expected to move in the videos. The assumptions are transformed into mathematical representations that are then used to predict where the target would be approximately located in a new frame. Such location information is exploited by a tracker to search for the target in an image area having a smaller size than the full frame. That smaller area – usually referred to as the searching area – permits the tracker to save computation and be more efficient since it does not need to process all the frame's pixels. The most common motion model employed in practice is the one considering the state predicted by the tracker in the previous frame as the location for the searching area in the next frame. For example, the coordinates of the center point of a bounding-box representation can be used as the coordinates of the center point of the searching area, while the width and height parameters can be scaled to obtain the size of the area (usually larger than the bounding-box). Overall, this kind of modeling is founded on the assumption that, in a standard frame-rate video (e.g. 20-30 frames-per-second), the target does not move much between consecutive frames. With this in mind, a quite large searching area placed at the previous target location would include the target object with high probability.

The last key factor to consider in visual tracking algorithms is the matching or similarity quantification operation. This is the step used to search for the template in the searching area. The process is generally implemented as an iterative procedure that compares the target model with an abstract model of the searching area at multiple locations. This produces a similarity map where higher values are associated with image regions in which template and searching area are most similar. To determine the new target position, it is sufficient to consider the coordinates of the peak value in such a map.

1.3.2 Traditional Methods

In this section, we discuss the most relevant approaches that have been exploited before the deep learning era and that today are referred to as traditional methods.

The oldest solutions for visual tracking were based on template matching [23, 24]. As their name suggests, these algorithms rely on a matching operation that is performed between the pixels of the template and those of the searching area. The template is shifted iteratively over the whole searching area. At each position, the matching operation computes a distance, for example, according to the sum of squared distances between the pixel values of the target and the part of the searching area having the same size. The overall procedure builds a similarity map between the template and the searching area, and the location having a shorter distance is retained as the new position for the target.

A competitive approach was based on mean shift algorithms [25]. This kinds of method were employed to implement an optimization strategy that induced the prediction of the new target's position towards the point resulting in the average similarity between template and searching area.

Other works exploited the information regarding the colors of the target and of the scene [26]. The underlying idea is to compute a similarity map between the template and the searching area based on their color similitude. This is achieved by using color histograms as appearance models and the technique of histogram backprojection [27] as matching operation.

Alternative algorithms used key-point [28] or part-based [29, 30] methods in order to develop more sophisticated target models capable of better addressing the deformations or partial occlusions of the target objects.

Later in time, the correlation filter approach gained popularity thanks to its fast processing speed [31, 32, 33, 34]. The idea behind this method is to perform a matching operation in which the target model is learned online as a filter capturing the best target's features. Such a filter is defined as a parameterized function with learnable parameters that are optimized during tracking. In the initialization process of the tracker, the strategy minimizes the difference between: the output of the correlation operation between the template pixels and the filter; a map of target occupancy obtained from the ground-truth bounding-box given in the first frame. The filter learned in the first frame is then updated at the successive frames in a similar fashion as described before but by using the predicted state of the tracker as ground-truth reference.

Even solutions based on the so-called tracking-by-detection [35, 36] gained considerable interest in the past. The concept is to learn online a classification function, acting as appearance model, capable of distinguishing image patches containing the appearance of the target from patches without it. After the classification model is learned, it can be used to classify image patches obtained from bounding-boxes whose position and width and height are sampled around the last known target's position. By retaining those boxes associated with high classification scores the new target position can be estimated.

1.3.3 Deep Learning-based Trackers

Just like the solutions for other computer vision problems, visual trackers leveraging deep learning demonstrated a large improvement in accuracy with respect to traditional methods. In this section, we briefly revise the most relevant works that implemented valid visual trackers based on deep neural networks.

The common trait behind the class of trackers based on deep learning is to overcome the difficulty of designing effective appearance models, motion models, or matching operations. The fundamental idea proposed by these tracking methods is to represent the aforementioned models by using deep neural networks and to train them on large-scale visual tracking-specific datasets [19, 37, 38, 39]. These are databases comprising thousands of videos whose frames have been manually labeled with the states of the targets (e.g. bounding-boxes) contained in the videos. Thanks to such an amount of information provided and the capability of deep networks of learning complex hierarchical functions, it is possible to learn automatically from pixel values many of the tools on which the community spent a lot of study in the past. Today, most of the visual tracking field switched from devising new handcrafted models to the design of new data-driven learning strategies and deep neural network architectures, in order to improve the performance of trackers (the step of training deep models on such big datasets is often referred to as offline learning).

One of the first solutions to employ deep learning methods in its tracking strategy has been the tracker known as MDNet [40]. This tracker addresses visual tracking by a tracking-by-detection paradigm. The idea is to train a deep neural network to distinguish image patches containing the target from those without. Such an ability is first acquired by extracting positive and negative samples considering the bounding-boxes contained in the large-scale video tracking datasets, and then solving a binary classification problem. Once trained, such discriminative capability is adjusted in every new video in which the tracker is applied. Positive and negative samples are extracted by the previously processed frames, and given that ground-truth information is not available during tracking, the polarity of the samples is obtained by considering the deep network's predicted scores. Once the training samples are obtained, the network's weights are fine-tuned by optimizing for a new binary classification.

Such a kind of methodology achieved outstanding results, but at the cost of reduced efficiency. This is due to the adaptation strategy – also referred to as online learning – performed while tracking the targets. Other types of solutions that gained popularity at the time of [40] ignored the employment of online learning and aimed to gain as much as tracking knowledge possible only by learning offline on large datasets. This has been achieved in the first place by deep regression trackers [41, 42] which proposed to learn, by solving a regression objective, a deep convolutional neural network to predict the shift between the bounding-boxes of two consecutive frames. Once optimized, such networks have been used to propagate iteratively the bounding-box given in the first frame to obtain the bounding-boxes in all the other frames without additional tuning. This resulted in trackers being extremely fast in their computation. Even though efficient, the accuracy of such kind of tracker was limited in many challenging situations. This issue has been partially mitigated by solutions that are referred to as siamese trackers [43, 44, 45, 46, 47]. This methodology improved the accuracy of deep regression trackers through the introduction of a new way of performing and learning tracking offline. In short, the proposed strategy aims to look for the features of the target template patch among the features abstracting a larger image in which the target is supposed to be present. Specifically, this operation is implemented as a cross-correlation between the former's and the latter's features and results in an activation map that highlights the region in which target and searching area the most similar. The features for both the target and searching areas are obtained through a deep convolutional network executed in the siamese fashion [48]. The overall pipeline is trained in an offline end-to-end manner where template and searching area pairs are extracted from annotated videos. The overall solution enables the network to learn how to produce the best image features that maximize the ability of tracking an object.

Following the success of reinforcement learning in many challenging problems [49, 16], different solutions explored the use of such techniques in the context of visual tracking [50, 51, 52, 53, 54]. The underlying idea is to model the tracking problem as a Markov Decision Process [55] and to learn a target-tracking policy parameterized by a deep network with optimization algorithms based on value functions [56], policy gradient [57], or actor-critic methods [58].

One of the most successful methodologies of the last years are discriminative correlation filters based on convolutional neural networks [59, 60, 61, 62]. The idea behind this approach is to use deep neural networks to implement effective functions able to discriminate effectively the target from the surrounding background by online adaptation. Upon this idea, different solutions have been introduced. The ECO tracker [60] proposed a strategy to exploit efficiently the features produced by deep networks. The ATOM tracker [61] introduced a particular neural network architecture to predict the estimated overlap between the predicted bounding-box and the actual target. The output of this model is exploited to define a maximization problem aimed to better optimize the tracker's weights online. Inspired by meta-learning [63], the DiMP tracker [62] introduced a deep network optimized for predicting the appearance model that best represents the target but that also results in the best form for an effective online adaptation during tracking.

Inspired by the success in natural language processing, the transformer architecture [64] is now receiving increased interest to tackle visual tracking problems [65, 66, 67]. In the context of this problem, the attention mechanism of such kind of data processing and learning methodology has been exploited to achieve the learning of more informative and discriminative features. This novelty additionally enhanced the performance of trackers, ultimately making them achieve new state-of-the-art results.

1.4 Motivations, Contributions, and Outline of this Thesis

Nowadays deep learning is the way-to-go for the implementation of strong visual tracking algorithms. Despite being so popular, the study of such methodologies for the visual object tracking problem is relatively at its early stages. This means that there are still many open issues that need to be addressed to fully comprehend the capabilities and potentialities of deep learning-based trackers. In this Thesis, we try to give an answer to some of these questions.

In the following of this section, we provide an overview about the problems tackled in this work and the relative contributions.

Learning Visual Tracking End-to-End by Deep Reinforcement Learning and an Expert Tracker

The Thesis begins by focusing on an issue that affected those methodologies that used reinforcement learning to optimize deep neural networks for visual tracking. Particularly, the first chapter deals with the two-stage learning procedure employed by such class of trackers. The contribution presented here substitutes the learning procedure with a single end-to-end learning strategy that makes the learning of a tracking policy easier and more effective. The proposed solution is based on recent trends in the reinforcement learning community that showed that demonstrations of an expert agent can be efficiently used to speed-up the process of policy learning. Considering the fact that many different algorithms have been proposed to track a generic object in videos, their execution on recent large-scale video datasets can produce a great amount of various tracking behaviors. Hence, by taking inspiration from such works we propose two novel trackers, A3CT, which exploits demonstrations of a state-of-the-art tracker to learn an effective tracking policy, and A3CTD, which takes advantage of the same expert tracker to correct its behavior during tracking. Through an extensive experimental validation on the most popular visual object tracking benchmarks, we will show that the proposed trackers compete with state-of-the-art solutions while running in real-time.

The contents of this Chapter have been published in:

- Matteo Dunnhofer, Niki Martinel, Gian Luca Foresti, Christian Micheloni, "Visual Tracking by Means of Deep Reinforcement Learning and an Expert Demonstrator", *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019;

- Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Cehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, Abdelrahman Eldesokey, Jani Kapyla, Gustavo Fernandez, ..., Matteo Dunnhofer, ..., "The Seventh Visual Object Tracking VOT2019 Challenge Results", *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.

Tracking and Learning by Trackers

The idea of exploiting other trackers in the tracking pipeline discussed in the previous Chapter leads us to the introduction of a new deep learning-based framework that explicitly considers other trackers as sources of information. This framework tries to unify application objectives such as fast processing algorithms, accurate online adaptation methods, and fusion of trackers, that were reasoned independently before. Our proposed solution presents a compact student model represented by a deep neural network and trained via the marriage of knowledge distillation and reinforcement learning. The first strategy allows to transfer and compress the tracking knowledge of other trackers. The second scheme enables the model to learn evaluation measures which are then exploited online. After the learning process is complete, the student can be ultimately used to build (i) a very fast single-shot tracker, (ii) a tracker with a simple and effective online adaptation mechanism, (iii) a tracker that performs fusion of other trackers. We will perform an extensive validation campaign which will reveal that the proposed algorithms compete with real-time state-of-the-art trackers.

The contents of this chapter have been published in:

- Matteo Dunnhofer, Niki Martinel, Christian Micheloni, "Tracking-by-Trackers with a Distilled and Reinforced Model", *Asian Conference on Computer Vision (ACCV)*, 2020;

- Matej Kristan, Ales Leonardis, Jirí Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Cehovin Zajc, Alan Lukezic, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernández, ..., Matteo Dunnhofer, ..., "The Eight Visual Object Tracking VOT2020 Challenge Results", *European Conference on Computer Vision Workshops (ECCVW)*, 2020.

Combining Trackers in the Long-Term Context

The previous Chapter will discuss the importance of building upon the knowledge of multiple and complementary trackers to achieve improved tracking performance. However, the solution presented there focused mainly on short-term tracking scenarios. Such a context has been studied a lot for the development of algorithms that perform tracker fusion, i.e. that combine the capabilities of underlying trackers. Despite such an extended interest, the long-term tracking setting has not been taken into consideration much. To overcome this problem, in this Chapter, we will explicitly consider such challenging tracking scenarios. We will present a deep learning methodology to fuse the characteristics of two complementary state-of-the-art trackers that achieves enhanced long-term tracking performance. Our strategy perceives whether the two trackers are following the object of interest through an online learned deep verification model. Such a target recognition strategy enables the activation of a decision strategy that selects the best performing tracker, as well as corrects the performance of the failing one. The proposed solution is compared with several baselines and it will be shown to beat the state-of-the-art on four popular long-term visual tracking benchmarks.

The contents of this Chapter have been partially published in

- Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Cehovin, Alan Lukežič, Ondrej Drbohlav, Jani Käpylä, Gustav Häger, Song Yan, Jinyu Yang, Zhongqun Zhang, Gustavo Fernández, ..., Matteo Dunnhofer, ..., "The Ninth Visual Object Tracking VOT2021 Challenge Results", *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021;

and are currently under review as:

- Matteo Dunnhofer, Christian Micheloni, "CoCoLoT: Combining Complementary Trackers in Long-Term Visual Tracking", *submitted to a conference*, 2022;
- Matteo Dunnhofer, Kristian Simonato, Christian Micheloni, "Combining Complementary Trackers for Enhanced Long-Term Visual Object Tracking", *submitted to a journal*, 2022.

Making Deep Learning Trackers Adapting to New Domains

Despite being very effective, deep learning models are subject to two issues among the others: domain shift and overfitting. Indeed, when trained on large-scale generic data and deployed for particular applications, deep neural networks suffer from the shift between the training and test data distributions. Moreover, such networks lose their generalization abilities if trained directly on small application datasets. Chapter 5 will focus on such problems in the context of deep learning-based tracking applications. Deep regression trackers will be targeted in the study because of their fast processing ability which makes them suitable for real-time applications. Despite the efficiency, the accuracy of such kind of trackers is inadequate in many domains due to the before mentioned issues. The solution that will be presented overcomes the issues by a domain adaptation strategy. This is the first methodology of such a kind developed for such a class of trackers and in the context of visual tracking. To reduce the labeling effort we will propose a weakly-supervised adaptation strategy based on the tracking-by-trackers framework introduced in Chapter 3. In this case, reinforcement learning is used to express weak supervision as a scalar application-dependent and temporally-delayed feedback. At the same time, knowledge distillation is employed to guarantee learning stability and to compress and transfer knowledge from more powerful but slower trackers. Extensive experiments on five different robotic vision domains will demonstrate the relevance of our methodology. Real-time speed will be achieved on embedded devices and on machines without GPUs, while accuracy will reach significant results.

The contents of this Chapter have been published in:

- Matteo Dunnhofer, Niki Martinel, Christian Micheloni, "Weakly-Supervised Domain Adaptation of Deep Regression Trackers via Reinforced Knowledge Distillation", *IEEE Robotics and Automation Letters*, 2021.

First Person Vision: A New Challenging Domain

Different application domains can provide different targets and motions which can have different impact on deep learning-based trackers, and the previous Chapter will discuss a solution to mitigate such issues for robotic vision domains. However, there are many other application domains in which it is important to use trackers. Among these, there is First Person Vision (FPV). This is a sub-field of computer vision devoted to the study and the development of algorithms to process images and videos acquired from cameras mounted on the head of a person. Such a setup permits to replicate our first person viewpoint in a computer. Tracking algorithms that follow the objects manipulated by the camera wearer can provide useful information to understand human-object interactions which is fundamental in this domain. Despite a few previous attempts to exploit trackers in first person vision solutions, a methodical analysis of the performance of state-of-the-art trackers in this domain is still missing. Considering that the visual tracking algorithms available in the computer vision literature have significantly improved their performance for a large variety of target objects and tracking scenarios but ignored this important context, Chapter 6 will present the first systematic study of single object tracking in FPV. Our study extensively analyses the performance of different visual trackers including those based on traditional methods and deep learning, and FPV-specific baseline trackers. The analysis is performed with respect to different aspects, new performance measures, and FPV-specific applications. This is achieved through the introduction of TREK-150, a novel benchmark dataset composed of 150 densely annotated video sequences. Our results will show that object tracking in FPV is challenging, and will suggest that more research efforts are needed for this problem so that tracking could benefit FPV tasks.

Part of the contents of this Chapter have been published in:

- Matteo Dunnhofer, Antonino Furnari, Giovanni Maria Farinella, Christian Micheloni, "Is First Person Vision Challenging for Object Tracking?", *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021;

while another part is currently under review as:

- Matteo Dunnhofer, Antonino Furnari, Giovanni Maria Farinella, Christian Micheloni, "Visual Object Tracking in First Person Vision", *submitted to a journal*, 2022.

From Tracking with Bounding-Boxes to Tracking with Segmentation Masks

All the studies that will be presented in the previous Chapters assume that bounding-boxes are employed to represent the targets' states. With the Chapter that follows we will try to move away from such a representation. As the visual object tracking field is moving towards binary segmentation masks to define objects more precisely, here we will present an extensive exploration of deep learning-based segmentation methods available in the computer vision community that can be conditioned on the target to be tracked. Such segmentation methods will be used to provide target segmentation after the output of bounding-box trackers. By this strategy, any bounding-box tracker can be transformed into a segmentation tracker. Our analysis will show that the proposed combination allows bounding-box trackers to compete with recently proposed segmentation trackers while performing quasi real-time.

The contents of this Chapter have been published in:

- Matteo Dunnhofer, Niki Martinel, Christian Micheloni, “An Exploration of Target-Conditioned Segmentation Methods for Visual Object Trackers”, *European Conference on Computer Vision Workshops (ECCVW)*, 2020.

Precise Tracking of the Knee Cartilage in Ultrasound Videos

The last Chapter will present a particular tracking problem in which precise information regarding the position and shape of the target in the form of a segmentation mask is required. The tracking of the knee cartilage during ultrasound-guided minimally invasive procedures is fundamental to avoid damaging this structure during such interventions. In this Chapter, the first computerized method to track, accurately and efficiently, the femoral condyle cartilage in ultrasound sequences acquired under several clinical conditions will be presented. Specifically, a new deep learning approach that combines a deep learning segmentation method with the siamese framework is introduced to track the cartilage in temporal and spatio-temporal sequences of 2D ultrasound images. Through extensive performance validation, we will demonstrate that our algorithm is able to track the femoral condyle cartilage with an accuracy that is comparable to experienced surgeons. It will be additionally shown that the proposed method outperforms state-of-the-art segmentation models and trackers in the localization of the cartilage. Given these outcomes, we claim that the proposed solution has the potential for ultrasound guidance in minimally invasive knee procedures.

The contents of this Chapter have been published in:

- Matteo Dunnhofer, Maria Antico, Fumio Sasazawa, Yu Takeda, Saskia Camps, Niki Martinel, Christian Micheloni, Gustavo Carneiro, Davide Fontanarosa, “Siam-U-Net: encoder-decoder siamese network for knee cartilage tracking in ultrasound images”, *Medical Image Analysis*, 2020.

2 Learning Visual Tracking End-to-End by Deep Reinforcement Learning and an Expert Tracker

Deep learning architectures such as convolutional neural networks (CNNs) [20] pre-trained for image classification showed to be effective in many current visual tracking methodologies [40, 41, 42, 54]. Due to their discriminative power, CNN-generated feature representations are widely used to search the target in the consecutive frames of a video. This kind of information has been initially exploited in classification or tracking-by-detection methods [40, 68]. Despite their accuracy, the most significant drawback of such methods is that they require computational demanding procedures to search for candidate targets in new frames. Furthermore, even strong CNN models may not be able to capture all possible variations of targets and need to be updated online during tracking. In these scenarios, the tracker shall understand the quality of its tracking process and the target's motion status, in order to take decisions that update efficiently its model. Solutions that implement such a mechanism achieve excellent results [40, 60, 50, 53], but their processing speed is often far from being real-time. Moreover, the problem of taking decisions online requires algorithms capable of deciding intelligently at the right moments.

To address these issues, tracking methodologies based on reinforcement learning (RL) have been recently proposed [52, 50, 51, 54, 53]. The idea behind such works is to treat aspects like target searching procedures or tracking status evaluation as sequential decision-making problems. In these settings, an artificial agent implemented as a deep network is trained to take optimal sequential decisions to solve a tracking related task. This step ultimately leads to the development of a strategy to track the target object. The aforementioned solutions achieve competitive performance with state-of-the-art methods, at the expense of complex and demanding online update procedures that slow tracking. More importantly, these learning methods are usually not end-to-end and hence require at least two training stages. Often, the first is defined as an initial supervised learning (SL) stage and only after a second learning step based on RL is followed for fine-tuning.

We argue that better performance can be obtained by incorporating SL into an RL framework to make the training end-to-end. We claim that tracking demonstrations of an expert tracker can be used to guide RL tracking agents. Furthermore, we propose to simplify the online update strategy by taking advantage of the expert during tracking, improving tracking speed. We will demonstrate that RL functions needed for training can be also used during tracking to exploit the performance of the expert tracker and to consequentially improve the tracking accuracy.

In particular, in this chapter we introduce the following contributions:

1. a real-time CNN-based tracker named A3CT which is trained via an end-to-end RL method that takes advantage of the demonstrations of a state-of-the-art tracker;

2. a real-time CNN-based tracker named A3CTD which uses the RL functions learned during training to improve performance by exploiting the expert during the tracking phase.

The proposed trackers are built on a deep regression network for tracking [41, 42] and are trained inside an on-policy Asynchronous Actor-Critic framework [69] that incorporates SL and expert demonstrations. A state-of-the-art tracking algorithm [43] is run on a large-scale tracking dataset [39] to obtain the demonstrations. Experiments show that the proposed A3CT and A3CTD trackers perform comparably with state-of-the-art methods on the GOT-10k test set [39], LaSOT [38], UAV123 [70], OTB-100 [71] and VOT [72, 73] benchmarks. The trackers achieve a processing speed of 90 FPS and 50 FPS respectively.

2.1 Related work

2.1.1 Deep RL

RL concerns methodologies to train artificial agents to solve interactive decision-making problems [55]. Recent trends in this field (e. g. [49, 56, 74, 16]) showed the successful combination of deep neural networks (DNNs) and RL algorithms (so-called seep RL) in the representation of models such as the value or policy functions. Among the existing approaches, off-policy strategies aim to learn the state or the state-action value functions, that give estimations about the expected future reward of states and actions [75, 49, 56]. The policy is then extracted by choosing greedily the actions that yield the highest function values. On the other hand, on-policy algorithms directly learn the policy by optimizing the DNN with respect to the expected future reward [57]. There exist then hybrid approaches, known as Actor-Critic [76], that maintain and optimize the model representations of both the policy and state value (or state-action value) functions.

All these methods however suffer of slow convergence, especially in cases where continuous or high-dimensional action spaces are considered. Recent solutions (e. g.

[77, 78, 79, 80, 81]) propose to use expert demonstrations to help and guide the learning process.

2.1.2 Visual Tracking

Visual tracking has received increasing interest thanks to the introduction of new benchmarks [71, 38, 70, 39] and challenges [82, 83, 72, 84, 73].

Thanks to their superior representation power, in recent years various approaches based on CNNs appeared [41, 42, 40, 60, 43, 44, 46]. Held et al. [41] and Gordon et al. [42] showed how deep regression CNNs could capture the target's motion. However, these methods are trained using SL which optimizes parameters for just local predictions. In contrast, we propose a RL-based training which optimizes the DNN's weights for the maximization of performance in future predictions. Nam et al. [40] proposed an online tracking-by-detection approach by using a pre-trained CNN for image classification. Similarly, Danelljan et al. [59, 60] proposed a discriminative correlation filter approach by integrating multi-resolution CNN features. These solutions obtained outstanding results w.r.t. the previous methodologies, however they are very computationally expensive and can run at just 1 and 6 FPS respectively. Currently, the approach based on the Siamese framework is getting significant attention for their well-balanced tracking accuracy and efficiency [43, 85, 44, 45, 86, 46]. These trackers formulate the visual tracking as a cross-correlation problem and are leveraging effectively from end-to-end learning of DNNs. However their performance is susceptible to visual distractors due to the non-incorporation of temporal information or online fine-tuning. Conversely to this, our tracker present the use of an LSTM [87] to model the temporal relation of target's appearance between frames.

2.1.3 Deep RL for Visual Tracking

Very recently, deep RL has started to be increasingly used to tackle the visual tracking problem. The first solution in this direction was the work of Yun et al. [50], which proposed an Action-Decision network to learn a policy for selecting a discrete number of actions to modify iteratively the bounding box in the previous frame. Huang et al. [88] used a Deep-Q-Network [56] to learn a policy for adaptively selecting efficient image features during the tracking process. In the work of [51], the tracker was modeled as an agent that takes decisions during tracking whether: to continue tracking with a state-of-the-art tracker or to re-initialize it; and to update or not the appearance model of the target object. In [52], authors used a variant of REINFORCE [57] to develop a template selection strategy, encouraging the tracking agent to choose, at every frame, the best template from a finite pool of candidate templates. In [53], authors presented a tracker which, at every time step, decides to shift the current bounding box while remaining on the same frame, to stop the shift process and move to the next frame, to update on-line the weights of the model or to re-initialize the tracker if the target is considered lost. Finally, [54] proposed to substitute the discrete action framework of [50] with continuous actions, thus performing just a single action at every frame.

All the presented methods include a pre-training step that uses SL to build a baseline policy or some other module used later by the tracking agents. Only after, RL is used to fine-tune such policies and modules. We take inspiration from RL methods that exploit expert demonstrations and we propose a novel end-to-end methodology based on on-policy Actor-Critic framework [69] to train a DNN capable of tracking generic objects in videos. We also demonstrate that the state value function learned during training, can be directly used to exploit the expert during tracking, in order to adjust wrong tracking behaviors and to consequentially improve the tracking accuracy.

2.2 Methodology

The key idea of this chapter is to take advantage of an expert tracker for training and tracking. RL and expert demonstrations are used to train a DNN which is then capable of tracking autonomously a generic target object in a video. The same network is also capable of evaluating its own performance and the one of the expert, thus exploiting the latter's knowledge in potential failure cases.

2.2.1 Problem Setting

In our setting, the tracking problem follows the definition of a Markov Decision Process (MDP). The tracker is treated as an artificial agent which interacts with an environment that is obtained as an MDP defined over a video. MDPs are a standard formulation for RL tasks and are composed of: a set of states $\mathcal{S}$; a set of actions $\mathcal{A}$; a state transition function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$; a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; and a discount value $\gamma \in \mathbb{R}$.

The interaction with the video, which we call an episode, happens through a temporal sequence of observations $s_1, s_2, \cdots, s_t$, actions $a_1, a_2, \cdots, a_t$ and rewards $r_1, r_2, \cdots, r_t$. In the t-th frame, the agent is provided with the state s_t and outputs the continuous action a_t which consists in the relative motion of the target object, i.e. it indicates how its bounding box, which is known in frame $t-1$, should move to enclose the target in the frame t. This approach is similar to the MDP formulation given by Chen et al. [54], however we propose different definitions for the states, actions and rewards.

Preliminaries. Given a dataset $\mathcal{D} = \{\mathcal{V}_0, \cdots, \mathcal{V}_{|\mathcal{D}|}\}$, we consider the j-th video

$$\mathcal{V}_j = \left\{ F_t \in \{0, \cdots, 255\}^{w \times h \times 3} \right\}_{t=0}^{T_j} \tag{2.1}$$

as a sequence of frames F_t. Let $b_t = [x_t, y_t, w_t, h_t]$ be the t-th bounding box defining the coordinates of the top left corner, and the width and height of the rectangle that contains the target object. At time $t-1$, given F_{t-1} and b_{t-1}, the goal of the tracker is to predict the bounding box b_t that best fits the target in the consecutive frame F_t.

State. Every state $s_t \in \mathcal{S}$ is defined as a pair of image patches obtained by cropping frames F_{t-1} and F_t using the bounding box b_{t-1}. Specifically, $s_t = \rho(F_{t-1}, F_t, b_{t-1}, k)$, where $\rho(\cdot)$ crops the frames F_{t-1}, F_t within the area of the bounding box $b'_{t-1} = [x'_{t-1}, y'_{t-1}, k \cdot w_{t-1}, k \cdot h_{t-1}]$ that has the same center coordinates of b_{t-1} but which width and height are scaled by k. With this function and by choosing $k > 1$, we can control the amount of additional image context information that is provided to the agent.

Actions and State Transition. Each action $a_t \in \mathcal{A}$ consists in a vector $a_t = [\Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t] \in [-1,1]^4$ which defines the relative horizontal and vertical translations ($\Delta x_t, \Delta y_t$, respectively) and width and height scale variations ($\Delta w_t, \Delta h_t$, respectively) that have to be applied to b_{t-1} to predict the bounding box b_t. This is obtained through $\psi : \mathcal{A} \times \mathbb{R}^4 \to \mathbb{R}^4$ such that

$$\psi(a_t, b_{t-1}) = \begin{cases} x_t = x_{t-1} + \Delta x_t \cdot w_{t-1} \\ y_t = y_{t-1} + \Delta y_t \cdot h_{t-1} \\ w_t = w_{t-1} + \Delta w_t \cdot w_{t-1} \\ h_t = h_{t-1} + \Delta h_t \cdot h_{t-1} \end{cases} \tag{2.2}$$

After performing the action a_t, the agent moves from the state s_t into the state s_{t+1} which is defined as the pair of cropped images obtained from the frames F_t and F_{t+1} using the bounding box b_t.

Reward. The reward function $r(s_t, a_t)$ expresses the quality of the action a_t taken at state s_t. Our reward definition is based on the Intersection-over-Union (IoU) metric computed between b_t and the ground-truth bounding box, denoted as g_t, i. e.., $\text{IoU}(b_t, g_t) = (b_t \cap g_t)/(b_t \cup g_t) \in [0,1]$. At every interaction step t, the reward is formally defined as

$$r(s_t, a_t) = \begin{cases} \omega\left(\text{IoU}(b_t, g_t)\right) \text{ if } \text{IoU}(b_t, g_t) \geq 0.5 \\ -1 \text{ otherwise} \end{cases} \tag{2.3}$$

with

$$\omega(z) = 2(\lfloor z \rfloor_{0.05}) - 1 \tag{2.4}$$

flooring to the closest 0.05 digit, then shifting the input range from $[0,1]$ to $[-1,1]$.

Expert Demonstrations. To guide the learning of our tracking agent we take advantage of the positive demonstrations of an expert tracker. Given $\mathcal{V}_j$, the bounding box prediction of the expert at time t is denoted as $b_t^{(d)}$. The demonstrations are obtained as sequences of triplets $\{(s_t^{(d)}, a_t^{(d)}, r_t^{(d)})\}_{t=0}^{T_j}$, each containing a state, an action and a reward, respectively. Precisely, we have that $s_t^{(d)} = \rho(F_{t-1}, F_t, b_{t-1}^{(d)}, k)$ and $a_t^{(d)} = [\Delta x_t^{(d)}, \Delta y_t^{(d)}, \Delta w_t^{(d)}, \Delta h_t^{(d)}]$, where its elements are obtained through $\phi : \mathbb{R}^4 \times \mathbb{R}^4 \to \mathcal{A}$, defined as

$$\phi(b_t^{(d)}, b_{t-1}^{(d)}) = \begin{cases} \Delta x_t^{(d)} = (x_t^{(d)} - x_{t-1}^{(d)})/w_{t-1}^{(d)} \\ \Delta y_t^{(d)} = (y_t^{(d)} - y_{t-1}^{(d)})/h_{t-1}^{(d)} \\ \Delta w_t^{(d)} = (w_t^{(d)} - w_{t-1}^{(d)})/w_{t-1}^{(d)} \\ \Delta h_t^{(d)} = (h_t^{(d)} - h_{t-1}^{(d)})/h_{t-1}^{(d)} \end{cases} \tag{2.5}$$

Rewards are calculated as $r_t^{(d)} = r(s_t^{(d)}, a_t^{(d)})$.

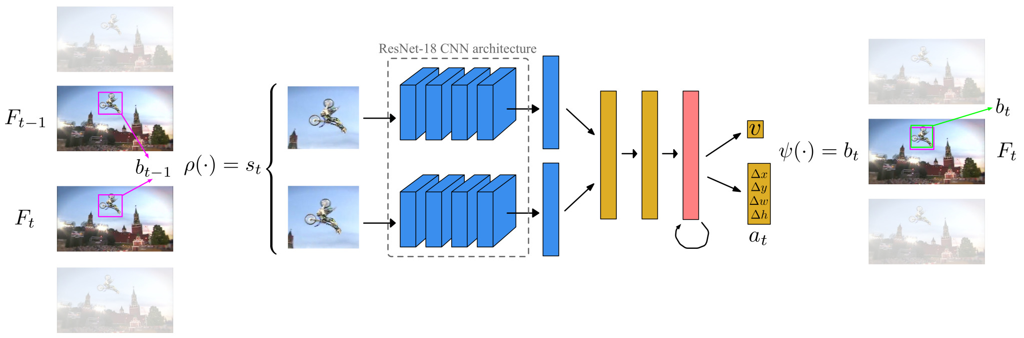


Figure 2.1: Visual representation of the interaction between the tracking agent and a video. Each pair of frames F_{t-1}, F_t is cropped by the function $\rho(\cdot)$ using the bounding box b_{t-1}. The obtained state s_t is fed to the agent's DNN which is composed by two branches of convolutional layers (the blue boxes) followed by, two fully-connected layers (rectangles in yellow), an LSTM layer (in light red) and two other fully connected layers for the prediction of v and the action a_t. Finally, the output bounding box b_t is built by the function $\psi(\cdot)$ which moves b_{t-1} by the relative shift a_t. (© 2019 IEEE)

2.2.2 Agent Architecture

Our tracking agent maintains representations of both the policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ and the state value function $v : \mathcal{S} \rightarrow \mathbb{R}$. This is done by using a DNN with parameters θ. In particular, we used a deep architecture that is similar to the one proposed by Gordon et al. [42].

The network gets as input two image patches. These pass through two convolutional branches that have the form of ResNet-18 CNN architecture [41] and which weights are pre-trained for image classification on the ImageNet dataset [19]. The two tensors of feature maps produced by the branches are first linearized, then concatenated together and finally fed to two consecutive fully connected layers with ReLU activations. After that, the features are inputted to an LSTM [87] RNN. Both the fully connected layers and the LSTM are composed of 512 neurons. The output of the LSTM is finally fed to two separate fully connected heads, one that outputs the action $a_t = \pi(s_t|\theta)$ and the other that outputs the value of the state, i.e. $v(s_t|\theta)$.

In Figure 2.1 a visual representation of the DNN architecture, together with the interaction process, is presented.

2.2.3 Training

The proposed DNN is trained solely off-line and in an end-to-end manner. The implemented training procedure is based on the on-policy A3C [69] RL framework. This method exploits P parallel and independent agents that interact with their own environments and that later use the gained experience to update asynchronously the weights θ which are shared among all agents. Indeed, each agent owns a copy θ' of the weights and this is synchronized with θ after every learning step. A3C is a standard algorithm in

RL, however it is not designed to take expert demonstrations into account. To overcome this limitation for our problem, we set up an A3C framework where a first half of the learning agents performs the traditional A3C learning, while the other half learns to imitate the actions of the expert tracker demonstrator in a supervised fashion.

Imitating Agents. Each imitating agent interacts with its environment by observing states, performing actions and receiving rewards just as standard A3C agents. Every t_{max} steps the agent updates the weights θ of the shared model with the gradients of the following loss function

$$\mathcal{L}_{imit} = \sum_{i=1}^{t_{max}} |\phi(b_t^{(d)}, b_{t-1}) - a_t| \cdot m_i. \tag{2.6}$$

which is the L1 loss between the actions performed by the learning agent and the actions that the expert tracker would take to move the agent's bounding box b_{t-1} into the expert's $b_t^{(d)}$. These absolute values are masked by the values $m_i \in \{0, 1\}$. Each of these is computed during the interaction and determines the situation in which the agent performed worse than demonstrator $(m_i = 1)$ or better $(m_i = 0)$. By optimizing the loss function 2.6, the weights θ are changed only if the agent's performance, in terms of received reward, is lower than the performance of the expert tracker. In simple words, the demonstrator is used to learn a baseline behavior on which the RL agent can build up its own tracking strategy, thus reducing the random exploration and consequentially speed up the learning process.

RL Agents. The training process performed by RL agents follows the standard structure proposed by Mnih et al. [69] for continuous control. Each agent interacts with the environment for a maximum of t_{max} steps. However, differently from the imitating agents, at each step t the RL agents sample actions from a normal distribution $\mathcal{N}(\mu, \sigma)$, where the mean is the predicted action, $\mu = \pi(s_t|\theta')$, and the standard deviation is obtained as $\sigma = |\pi(s_t) - \phi(g_t, b_{t-1})|$ (which is the absolute value of the difference between the agent's action and the action that obtains, by shifting b_{t-1}, the ground-truth bounding box g_t). Intuitively, σ shrinks $\mathcal{N}$ when the action a_t is close to the ground-truth action $\phi(g_t, b_{t-1})$, thus reducing the chance of choosing potential wrong actions when approaching the correct one. On the other hand, when the action a_t is far from $\phi(g_t, b_{t-1})$, σ takes a greater value, spreading $\mathcal{N}$. This allows the agent to explore more the environment and discover potential good actions.

Curriculum Strategy. In addition to the guiding process done by the imitating learners using the expert demonstrations, we designed a curriculum learning strategy [89] to further facilitate the training. In a similar way as proposed by [79], we built a curriculum based on the performance of the learning agents w.r.t. to the expert demonstrator. In particular, after terminating each episode, a success counter is increased if the agent performs better than the expert in that episode, i.e. if the former's cumulative reward, received up to $\widehat{T}_j$, is greater or equal to the one obtained by the latter. In formal terms, the success counter is updated if the following holds

$$\sum_{i=1}^{\widehat{T}_j} r_i \geq \sum_{i=1}^{\widehat{T}_j} r_i^{(d)}. \tag{2.7}$$

The counter update is done by testing agents that interact with the sequences by performing $\pi(s_t|\theta')$ using a local copy θ' of the shared weights. The terminal episode index $\widehat{T}_j$ is successively increased during the training procedure by a central process which checks if the ratio, between the number of episodes in which the learning agent performs better than the demonstrator and the total number of episodes terminated, is above the threshold τ. With this learning setting, we ensure that at every augmentation of $\widehat{T}_j$ the agents face a simpler learning problem where they are likely to succeed and in a shorter time, since they have already developed a tracking policy that, up to $\widehat{T}_j - 1$, is at least good as the one of the expert.

2.2.4 Tracking at Test Time

Despite the fact that the proposed tracker is trained by taking advantage of an expert's knowledge, our tracker develops a tracking ability that can be exploited independently from the tracking strategy used by the demonstrator. Nevertheless, it is possible to take advantage of the expert's tracking performance also during the tracking phase. Therefore we set up two tracking strategies, the first one that tries to track autonomously the target object and we refer it as A3CT, and the second one that takes advantage of the demonstrator's knowledge also during tracking and that we name A3CTD.

A3CT. In this setting, A3CT is applied straight away on an arbitrary sequence. Each tracking sequence $\mathcal{V}_j$, with target object outlined by g_0, is considered as the MDP described in section 2.2.1. The tracker computes states s_t from frames F_t, performs actions as by means of the learned policy $a_t = \pi(s_t|\theta)$ which are used to output the bounding boxes $b_t = \phi(a_t, b_{t-1})$. At the beginning, $b_0 := g_0$.

No online update of the network's weights nor of the LSTM's hidden state are performed.

A3CTD. During training, the tracking agent learns both the policy $\pi(s_t|\theta)$ and the value function $v(s_t|\theta)$. $v(\cdot)$ is a function that predicts the reward that the agent expects to receive from the current state s_t to the end of the sequence. Since our reward definition is a direct measure of the IoU between the predictions of the agent and the ground-truth bounding boxes, $v(s_t|\theta)$ gives an estimate of the total amount of IoU that the tracker expects to obtain from state s_t on wards. This function can be exploited as a performance evaluation for both our tracker and the expert demonstrator. In particular, at each time step t, $\widehat{R} = v(s_t|\theta)$ and $\widehat{R}^{(d)} = v(s_t^{(d)}|\theta)$ are obtained as the evaluation for A3CTD and the expert tracker respectively. The expert state $s_t^{(d)}$ is obtained by cropping frames F_{t-1}, F_t using its previous prediction $b_{t-1}^{(d)}$. By comparing $\widehat{R}$ and $\widehat{R}^{(d)}$, our strategy decides if to output the bounding box of A3CTD or the bounding box produced by the expert tracker. More formally, if $\widehat{R} \geq \widehat{R}^{(d)}$ then the tracker outputs $b_t := \phi(a_t, b_{t-1})$ otherwise it outputs $b_t := b_t^{(d)}$.

2.2.5 Implementation Details

In this section we report the results of the hyperparameters search which led to the best performance.

Before being fed to the DNN, the image crops that forms the MDP states are resized to $[128 \times 128 \times 3]$ pixels and standardized, per channel, by subtracting the mean and dividing by the standard deviation calculated on the ImageNet dataset [19]. The dilating factor k is set to 1.5.

A total number of $P = 16$ training agents was used. The discount factor γ was set to 1. The length of the rollout was defined in $t_{max} = 5$ steps. τ was set to 0.25. The model was trained for 40000 episodes using the Adam optimizer [90]. The learning rate for both imitating and training agents was set to 10^{-6}. A weight decay of 10^{-4} was also added to the L1 loss of the imitating agents as regulatory term.

Training and experiments have been conducted running our Python code with the PyTorch [91] machine learning library on an Intel Xeon E5-2690 v4 @ 2.60GHz CPU with 320 GB of RAM, four NVIDIA TITAN V GPUs and an NVIDIA TITAN Xp GPU each with 12 GB of memory. The training took around 4 days. In the evaluation of trackers' speed, we ignore disk read times since they do not dependent on the tracking algorithm.

Expert Tracker. The role of expert tracker was assigned to SiamFC [43]. The choice was motivated by the fact that this solution is nowadays an established methodology in the visual tracking panorama, and it shows great balance in results across many different benchmarks. In particular, SiamFC has currently one of the best performance on the public leader-board of the GOT-10k test set. Additionally, the source code was publicly available.

To obtain tracking demonstrations, we ran SiamFC on the training set of GOT-10k dataset [39]. The implemented SiamFC was trained on the ImageNet VID dataset [19]. This is an important aspect because, to train our tracking agent, we want examples of the tracker's real behaviour, that must be obtained on never seen before sequences. Moreover, demonstrations that are clearly useful are needed. So, of all the trajectories produced, we retained just the ones considered positive, i.e. the trajectories that satisfy $\text{IoU}(b_t^{(d)}, g_t) > 0.5$ for all $t \in \{1, \ldots, T_j\}$. All the others were discarded.

Training Dataset. To train A3CT and A3CTD we leveraged of the training set of the GOT-10k dataset [39]. This is a large-scale dataset containing 9335 training videos, 180 validation videos and other 180 videos for testing. In total, this dataset provides 1.5M bounding boxes that identify 10k different target objects. The latters belong to 563 distinct object classes. The actual number of training sequences we used is however inferior. In fact, just the videos which obtained a positive demonstration from the expert tracker were employed for training. Furthermore, as we aimed to take part to the VOT 2019 challenge, we removed 1000 sequences from the training set. These overlapped

Table 2.1: State-of-the-art comparison on the GOT-10k test set in terms of average overlap (AO), and success rates (SR) with overlap thresholds 0.5 and 0.75. Except for ATOM, both versions of our approach outperform the previous methods in all three measures. (© 2019 IEEE)

	KCF [32]	MDNet [40]	ECO [60]	CCOT [59]	GOTURN [41]	SiamFC [43]	SiamFCv2 [93]	ATOM [61]	**A3CT**	**A3CTD**
AO	0.203	0.299	0.316	0.325	0.347	0.348	0.374	0.556	**0.415**	**0.425**
$\mathrm{SR}_{0.50}$	0.177	0.303	0.309	0.328	0.375	0.353	0.404	0.634	**0.477**	**0.495**
$\mathrm{SR}_{0.75}$	0.065	0.099	0.111	0.107	0.124	0.098	0.144	0.402	**0.212**	**0.205**

with the pool of videos used by the VOT committee for evaluation. After these pruning steps, the total amount of training samples, $|\mathcal{D}|$, resulted in 1782 videos.

2.3 Experiments

In this section we report the experimental setup and we discuss the results, obtained by the proposed trackers A3CT and A3CTD, on the benchmarks GOT-10k [39], LaSOT [38], UAV123 [70], OTB-100 [71], VOT2018 [73] and VOT2019 [92].

2.3.1 Comparison with the State-of-the-Art

GOT-10k Test Set. The GOT-10k [39] test set comprises 180 videos. Target objects belong to 84 different classes and 32 forms of object motion are present. To ensure a fair evaluation, the trackers that are evaluated on this benchmark are forbidden from using external datasets for training. The evaluation protocol proposed by the authors is the one-pass evaluation (OPE) [71]. The metrics used are the average overlap (AO) and the success rates (SR) with overlap thresholds 0.5 and 0.75.

In Table 2.1 we report the results of A3CT and A3CTD against the state-of-the-art. A3CT outperforms the state-of-the-art trackers which, at the time of writing, appear on the GOT-10k test set leaderboard. In particular, it has a better tracking performance w.r.t. to the demonstrator tracker SiamFC [43], with a performance gain of 6.7% and in AO, 12.4% in $\mathrm{SR}_{0.50}$, and 11.4% in $\mathrm{SR}_{0.75}$. A3CTD increases additionally the performance of A3CT, with an improvement of 1% in AO, 1.8 in $\mathrm{SR}_{0.50}$ but with a loss of 0.7% in $\mathrm{SR}_{0.75}$. We perform worse than ATOM [61], however we remark that these results are obtained considering just 1782 of the 9335 sequences (19%) contained in the GOT-10k training set.

OTB-100. The OTB-100 [71] benchmark is a set of 100 challenging videos and it is widely used in the tracking literature. The standard evaluation procedure for this dataset is the OPE method and the metrics used are the success plot and the precision plot. The Area Under the Curve (AUC) of these curves are referred as success score (SS) and precision scores (PS) respectively.

In Table 2.2 we report the success and and precision scores against state-of-the-art solutions. On this benchmark, A3CT and A3CTD have lower performance than ECO [60], MDNet [40], SiamRPN++ [45] and the expert SiamFC [43]. However, A3CT still

Table 2.2: State-of-the-art comparison on the OTB-100 benchmark in terms of success score (SS) and precision score (PS). (© 2019 IEEE)

	GOTURN [41]	RE3 [42]	KCF [32]	SiamFC [43]	ACT [54]	MDNet [40]	ECO [60]	SiamRPN++ [45]	**A3CT**	**A3CTD**
SS	0.395	0.464	0.477	0.575	0.625	0.677	0.691	0.696	**0.419**	**0.535**
PS	0.534	0.582	0.693	0.762	0.859	0.909	0.910	0.914	**0.568**	**0.717**

Table 2.3: State-of-the-art comparison on the LaSOT benchmark in terms of success score (SS) and precision score (PS). (© 2019 IEEE)

	KCF [32]	GOTURN [41]	ECO [60]	RE3 [42]	SiamFC [43]	MDNet [40]	SiamRPN++ [45]	**A3CT**	**A3CTD**
SS	0.178	0.214	0.324	0.325	0.336	0.397	0.496	**0.306**	**0.415**
PS	0.166	0.175	0.301	0.301	0.339	0.373	-	**0.246**	**0.368**

performs better than GOTURN [41]. A3CTD instead outperforms RE3 [42] and KCF [32], with a 5.8-7.1% performance gain in SS and 1.8-13.5% in PS. In this setting, the help of the expert tracker is crucial to improve the results of A3CT, which sees an improvement of 11.6% in SS and 14.9% in PS.

LaSOT. We performed evaluations of A3CT and A3CTD performance on the test set of LaSOT benchmark [38]. This dataset is composed of 280 videos with a total of more than 650k frames and an average sequence length of 2500 frames. To evaluate our tracker, we use the same methodology and metrics used for the OTB-100 experiments.

In Table 2.3 we present the results against state-of-the-art trackers. In this setting, in terms of SS A3CT performs comparably to ECO [60] and RE3 [42] but much better than GOTURN [41]. Also in this case, the aid of the expert tracker is crucial, which results in a increment of 10.9% in SS and of 12.2% in PS. A3CTD so outperforms the expert SiamFC [43] in SS by 7.9% and MDNet [40] by 1.8%. Both our trackers are however weaker than SiamRPN++ [45].

UAV123. The UAV123 [70] is a benchmark composed of 123 videos acquired from low-altitude UAVs. The dataset is inherently different from traditional visual tracking benchmarks like OTB and VOT, since it offers sequences with an aerial point of view. To evaluate our trackers, we use the same methodology and metrics used for the OTB-100 experiments.

In Table 2.4 we present the scores against state-of-the-art trackers. A3CT performs 14%, 8.2% and 5.6% better, in terms of SS, than KCF [32], GOTURN [41] and ACT [54] respectively. A3CTD has a 9.4% SS and a 13.2% PS improvements than A3CT and these lead to outperform SiamFC [43], ECO [60] and MDNet [40] with a gain of, respectively, 4.2%, 4%, 3.7% in SS and 2.4%, 1.3%, 0.7% in PS.

VOT Benchmarks. The VOT benchmarks are datasets used in the annual VOT tracking competition. These sets change year by year, introducing challenging tracking scenarios and increasing the difficulty of the task. Within the framework used by the VOT committee, trackers are evaluated based on Expected Average Overlap (EAO),

Table 2.4: State-of-the-art comparison on the UAV123 benchmark in terms of success score (SS) and precision score (PS). (© 2019 IEEE)

	KCF [32]	GOTURN [41]	ACT [54]	RE3 [42]	SiamFC [43]	ECO [60]	MDNet [40]	SiamRPN++ [45]	**A3CT**	**A3CTD**
SS	0.331	0.389	0.415	0.514	0.523	0.525	0.528	0.613	**0.471**	**0.565**
PS	0.523	0.548	0.636	0.667	0.730	0.741	0.747	0.807	**0.622**	**0.754**

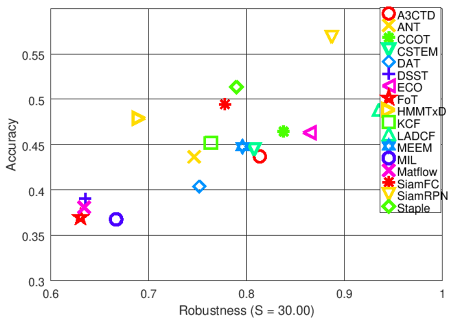


Figure 2.2: Accuracy-Robustness plot against some of the VOT2018 [73] competitors. (© 2019 IEEE)

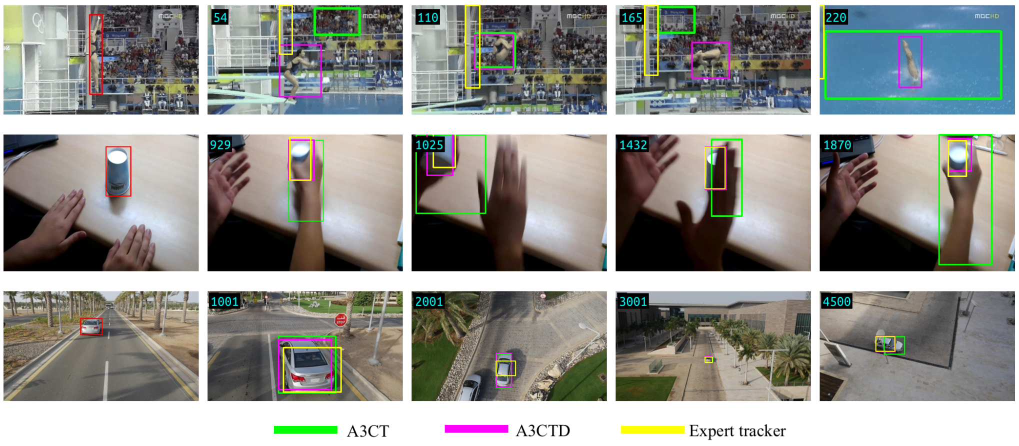


Figure 2.3: Qualitative examples of A3CT and A3CTD performance. (© 2019 IEEE)

Accuracy (A) and Robustness (R) [94]. We performed experiments on the test sets of VOT2018 and VOT2019 challenges. Both two benchmarks provide 60 (non completely overlapping) challenging videos.

In Figure 2.2 we present the Accuracy-Robustness plot including A3CTD's performance in comparison with some of the partecipants to the VOT-2018 challenge. A3CTD achieves an EAO of 0.1847, an accuracy of 0.4536 while it failed (i.e. the IoU with the

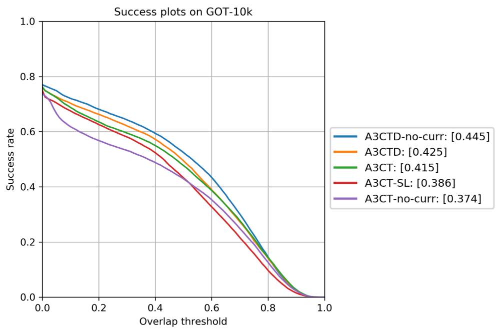


Figure 2.4: Success plot for the ablation study of A3CT and A3CTD. (© 2019 IEEE)

ground-truth becomes zero) 34.89 times. Our method perform definitely worse than the best solutions LADCF [95], SiamRPN [44] and ECO [60] that achieved an EAO of 0.3889, 0.3837 and 0.2809 respectively. A3CTD's performance is however comparable to the one of SiamFC [43], which achieved an EAO of 0.1875.

We submitted the A3CTD tracker to the VOT2019 challenge [92], since it resulted in the best performance generally. It achieved an EAO of 0.1652 baseline experiments resulting in the 41st position. The overlap in the baseline experiment resulted in 0.4510.

2.3.2 Ablation Study

To assess the validity of all the features of our proposed solution we performed an ablation study on the GOT-10k test set. In particular, we ran experiments where we trained A3CT and A3CTD without the curriculum strategy (A3CT-no-curr and A3CTD-no-curr respectively) and A3CT with just imitating agents (A3CT-SL). In Figure 2.4 we report the success plot with the comparison of the different models involved. A3CT-SL performs worse than A3CT, suggesting that the use of RL agents is crucial to improve the baseline behaviour learned by the imitating agents. Moreover, since the state value function is learned by RL agents, this setup does not allow to exploit the demonstrator in the tracking phase. A3CT-no-curr performs comparably to A3CT-SL, 4.1% lower than A3CT. The curriculum learning strategy allows the tracking agent to learn a more precise tracking policy. Interestingly, A3CTD-no-curr outperforms A3CTD by 2%. We believe that the increased length of the sequences during training allows the learning of a more accurate state value function, which is then able to make better predictions about the future behaviours of A3CT and the expert tracker. However, we chose A3CT and A3CTD as our final solution because of their lower difference in performance.

In terms of processing speed, A3CT runs at 90 FPS while A3CTD runs at 50 FPS.

Finally, in Figure 2.3 we present some qualitative examples of the tracking performance of A3CT and A3CTD.

2.4 Conclusions

This chapter dealt with the problem of simplifying the optimization procedure of deep models employed in RL-based visual trackers. Thanks to the availability of a great amount of visual tracking algorithms and inspired by the recent trends in RL, we proposed two novel trackers that are built on a deep regression network [41, 42] optimized with a new end-to-end learning strategy. The state-of-the-art tracking algorithm SiamFC [43] was executed on the large-scale tracking dataset GOT-10k [39] to obtain expert demonstrations. The proposed network was then trained inside an RL on-policy asynchronous Actor-Critic framework [69] that incorporated parallel SL agents. Experiments showed that the proposed A3CT and A3CTD trackers outperform state-of-the-art methods on the GOT-10k [39], LaSOT [38], and UAV123 [70] benchmarks, and perform comparably with the state-of-the-art on OTB-2015 [71] and VOT benchmarks [73]. Moreover, A3CT and A3CTD achieved a processing speed of 90 and 50 FPS respectively and thus are suitable for real-time applications.

3 Tracking and Learning by Trackers

In the previous chapter we presented a methodology to improve the learning of trackers that exploits reinforcement learning (RL). Our solution was made possible thanks to the introduction of a tracker that is considered as an expert and from which tracking can be learnt. In the study appearing in this chapter we extend such an idea. We present a new visual tracking framework in which trackers are considered as fundamental building blocks, both for the learning phase and for the actual tracking phase. Our idea is based on the realization that nowadays many different principles are available to track objects in videos as discussed in depth in the introductory chapter. Based on such principles, a large number of algorithms has been produced so far. Thus, one can imagine that different trackers incorporate different knowledge that may constitute a valuable resource to leverage on during tracking.

By proposing the currently available trackers, we noticed that the community focused on single aspects of the visual tracking problem in a shortsighted way. In particular, an high processing speed was pursued by algorithms like correlation filters [32] or methods such as offline siamese CNNs [41, 42, 44, 96, 97, 98]. Improved performance was aimed by online target adaptation solutions [40, 60, 61, 62], while other approaches tried to take advantage of the output produced by multiple trackers [99, 100, 101]. We believe that all these capabilities should belong to an optimal tracking solution. However, the community currently lacks a general solution to achieve them jointly. In this view, such an algorithm should provide mechanisms to (i) apply decision-making strategies to combine the outputs of multiple trackers, (ii) implement simple and effective online adaptation procedures, (iii) track at high speed.

The knowledge distillation (KD) framework [102] was introduced in the deep learning panorama as paradigm for, among the many [103, 104], knowledge transferring between models [105] and model compression [106]. The idea boils down to a student model learning from one or more teacher model. Teachers explicit their knowledge through demonstrations on a never seen before transfer set. Through specific loss functions, the student learns a task by matching the teachers' output and the ground-truth labels at the same time. As visual tracking requires fast and accurate methods, KD can be a valuable tool to transfer the tracking ability of more accurate teacher trackers to more compact thus faster student ones. However, the standard setup of KD does not provide methods to exploit teachers online but just offline. This makes this methodology unsuitable for tracking, which has been shown to benefit from both offline and online methods [50, 54, 60, 61, 62]. In contrast to such an issue, RL techniques offer established methodologies to optimize not only policies but also policy evaluation functions [75, 58], which can be used to derive decision strategies. RL also gives the opportunity to optimize models for arbitrary and non-differentiable performance metrics. Hence, more tracking oriented objectives that could enhance the learning process can be defined with such tools.

For the aforementioned motivations, the contribution of this chapter is a novel tracking framework that aims to unify and achieve the goals (i), (ii) and (iii). Our proposed methodology is based on the consideration that visual tracking algorithms contain beneficial tracking knowledge. In particular, we treat off-the-shelf trackers as teachers, and we train a student model to exploit them both offline and online. Specifically, the student is trained offline via an effective strategy that combines KD and RL to compress, enhance, and evaluate the tracking policy of the teachers. Then, at test time, our tracking algorithm uses student and teachers interchangeably in different configurations depending on the application needs. We will show how to exploit them in three modalities which result in, respectively, (i) an accurate and robust setup that exploits multiple tracking teachers to perform tracker fusion (TRASFUST), (ii) a setup that uses a single teacher to implement a simple online adaptation mechanism (TRAST), and (iii) a fast processing setup which compresses the teacher's knowledge in its tracking policy (TRAS). Through extensive evaluation procedures, it will be demonstrated that our tracking solution competes with different state-of-the-art trackers while performing in real-time.

3.1 Related Work

We now discuss the tracking algorithms most similar to the proposed three modalities, as well as comparable learning strategies based on KD and RL that have been developed for visual tracking or for other computer vision tasks.

3.1.1 Visual Tracking

The network architecture that implements our proposed student model takes inspiration from GOTURN [41] and RE3 [42]. These regression-based CNNs were shown to capture the target's motion while performing at more than 100 FPS. However, the learning strategy employed by such methods optimizes just for bounding-box coordinate difference. Moreover, a great amount of data is needed to make these models accurate. In contrast, our KD-RL-based method offers optimization for bounding-box overlap maximization, and extracts previously acquired knowledge from other trackers thus requiring less data for training.

Online adaptation methods like discriminative model learning [40] or discriminative correlation filters have been studied extensively [60, 61, 62, 107] to improve tracking accuracy. These procedures are generally very complex because they require particular assumptions and careful design. Our proposed online update strategy exploits an off-the-shelf tracker to correct the performance of the student model, while posing any constraint on such tracker if not a bounding-box output. Thus, our strategy gives the freedom of developing disparate adaptation procedures.

Currently available tracker fusion methods use trackers in the form of discriminative trackers [99], CNN feature layers [108], correlation filters [109] or out-of-the-box tracking algorithms [100]. The main issue is that such methods work just online and do not take advantage of the great amount of offline knowledge that trackers can provide. Furthermore, such solutions do not present mechanisms to track objects without the underlined trackers. Our framework addresses these issues thanks to the student model which compresses the offline knowledge of other trackers to perform tracking autonomously.

3.1.2 KD and RL

KD techniques have been used for transferring knowledge between teacher and student models [102], where the supervised learning setting was employed more [103, 104, 105] than the setup that uses RL [110]. In the context of computer vision, KD was employed for image recognition [111], object detection [112], semantic segmentation [113], or person search [114]. In the visual tracking panorama, KD was explored in [115, 116] to compress, CNN representations for correlation filter trackers and siamese network architectures. Despite the good results, these works offer methods involving teachers specifically designed as correlation filter and siamese trackers. Hence, they can not extract knowledge from generic-approach visual trackers like we propose in this study. Moreover, to the best of our knowledge, no method mixing KD and RL is currently present in the computer vision literature.

Our learning procedure is also related to the strategies that use deep RL to learn tracking policies [50, 54, 117]. Our formulation shares some characteristics with such methods in the Markov Decision Process (MDP) but proposes different definitions for states, actions, and reward function. Moreover, our proposed learning algorithm is novel as none of the presented methods leverages on teachers to learn the tracking policy. On this point, the study showcased in the previous chapter contains the most similar learning method. However, such a method uses just a single tracker for learning, while in this study we propose a deeper analysis of the methodology and a generalization to multiple trackers.

3.2 Methodology

The key point of this study is the exploitation of off-the-shelf tracking algorithms to develop efficient trackers under different perspectives such as accuracy and robustness, target adaptation, and speed. In particular, as both offline and online strategies have been shown to be necessary for visual tracking [50, 54, 61, 62], we present a novel tracking framework in which a student model first learns from teacher trackers offline and then exploits them online. To implement this idea, the KD and RL techniques are joined.

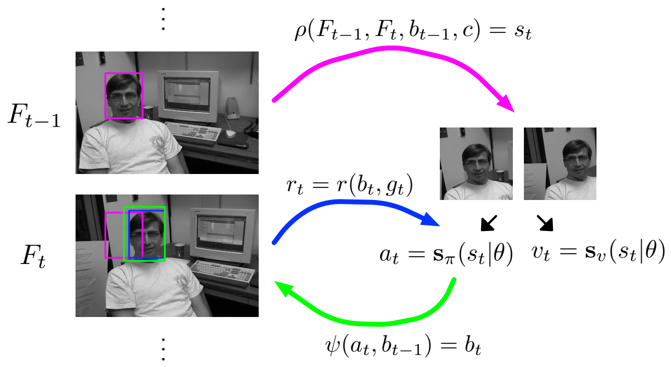

$\rho(F_{t-1}, F_t, b_{t-1}, c) = s_t$
F_{t-1}
F_t
$r_t = r(b_t, g_t)$
$a_t = \mathbf{s}_\pi(s_t|\theta)$
$v_t = \mathbf{s}_v(s_t|\theta)$
$\psi(a_t, b_{t-1}) = b_t$

Figure 3.1: Visual representation of the MDP employed in our framework. Each pair of frames F_{t-1}, F_t is cropped by $\rho(F_{t-1}, F_t, b_{t-1}, c)$ that uses the bounding-box b_{t-1} (magenta boxes) and a context factor c. The obtained state s_t is fed to the student that performs the action a_t and estimates the tracking performance v_t. The output b_t (green box) is built through $\psi(a_t, b_t)$. a_t is then rewarded by the overlap-based measure $r(b_t, b_t^{(g)})$ with $b_t^{(g)}$ being the ground-truth (blue box).

KD is used for transferring the tracking knowledge from the off-the-self trackers to the compressed student model. But since the standard KD setup does not offer a way to exploit teachers online, we propose to augment it with an RL optimization objective. RL techniques deliver unified optimization strategies to directly maximize a desired performance measure and to predict its expectation. We use the latter advantage as base for an online teacher evaluation and selection strategy. Instead, the former benefit is employed to improve the distilled tracking policy by means of a bounding-box overlap based objective.

We begin this section by introducing some preliminary concepts. We then give the definition of the MDP we used to model the visual tracking procedure. Next, the offline learning procedure is presented and the student architecture is described. Finally, we will describe the proposed tracking strategies in which student and teacher are employed.

3.2.1 Preliminaries

We consider a video $\mathcal{V} = \{F_t \in \mathcal{I}\}_{t=0}^{T}$ as a sequence of frames F_t, where $\mathcal{I} = \{0, \cdots, 255\}^{w \times h \times 3}$ is the space of RGB images and $T \in \mathbb{N}$ denotes the number of frames in the video. Let $b_t = [x_t, y_t, w_t, h_t] \in \mathbb{R}^4$ be the t-th bounding-box defining the coordinates of the top left corner, and the width and height of the rectangle containing the target. At time t, given the current frame F_t, the goal of a tracker is to predict b_t that best fits the target in F_t. At the first frame F_0, the tracker is initialized with the ground-truth bounding-box $b_0^{(g)}$ which outlines the target to be tracked. For training, we consider the transfer set $\mathcal{D} = \{\mathcal{V}_j\}_{j=0}^{|\mathcal{D}|}$ as a set of videos each T_j-frames long. We denote the tracking student model with $\mathbf{s}$ and a tracking teacher algorithm with $\mathbf{t}$.

3.2.2 Visual Tracking as an MDP

In our framework, the video processing procedure to track a target is considered as an MDP defined over the j-th video $\mathcal{V}_j$ of $\mathcal{D}$. The student $\mathbf{s}$ is treated as an artificial agent which interacts with such an MDP. The interaction happens through a temporal sequence of states $s_1, s_2, \cdots, s_t \in \mathcal{S}$ and actions $a_1, a_2, \cdots, a_t \in \mathcal{A}$. During the learning phase, the interaction includes the rewards $r_1, r_2, \cdots, r_t \in [-1, 1]$. In the t-th frame, the student is provided with the state s_t and outputs the continuous action a_t which is rewarded by the measure of its quality r_t while interacting for learning. We refer to this interaction process as episode E_j, whose dynamics are defined by the MDP $M_j = (\mathcal{S}, \mathcal{A}, r)$. A schematic representation of the MDP is proposed in Figure 3.1.

States. Every state $s_t \in \mathcal{S}$ is defined as a pair of image patches obtained by cropping frames F_{t-1} and F_t using the previously known bounding-box b_{t-1}. Specifically, $s_t = \rho(F_{t-1}, F_t, b_{t-1}, c)$, where $\rho(\cdot)$ is a function that crops the frames F_{t-1}, F_t within the area of the bounding-box $b'_{t-1} = [x'_{t-1}, y'_{t-1}, c \cdot w_{t-1}, c \cdot h_{t-1}]$ that has the same center coordinates of b_{t-1} but whose width and height are scaled by the factor c. By selecting $c > 1$, we can control the amount of additional image context information provided to the student.

Actions and State Transition. Each action $a_t \in \mathcal{A}$ consists of a vector $a_t = [\Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t] \in [-1, 1]^4$. This defines the relative horizontal and vertical translations ($\Delta x_t, \Delta y_t$, respectively) and width and height scale variations ($\Delta w_t, \Delta h_t$, respectively) that have to be applied to b_{t-1} to obtain b_t. Such a transformation is implemented through the function $\psi : \mathcal{A} \times \mathbb{R}^4 \to \mathbb{R}^4$ defined as

$$\psi(a_t, b_{t-1}) = \begin{cases} x_t = x_{t-1} + \Delta x_t \cdot w_{t-1} \\ y_t = y_{t-1} + \Delta y_t \cdot h_{t-1} \\ w_t = w_{t-1} + \Delta w_t \cdot w_{t-1} \\ h_t = h_{t-1} + \Delta h_t \cdot h_{t-1}. \end{cases} \tag{3.1}$$

After taking a_t, the student moves to the next frame F_{t+1} to extract the new state s_{t+1} defined as $s_{t+1} = \rho(F_t, F_{t+1}, b_t, c)$.

With the proposed definitions for $\mathcal{S}$ and $\mathcal{A}$, we set the student to predict the relative motion of the target between the two consecutive image patches. In other words, $\mathbf{s}$ indicates how the target bounding-box, known at frame $t-1$, should move and scale to enclose the target at frame t.

Reward. The reward function expresses the quality of the action a_t taken at state s_t, and it is used to feedback the student just during the learning phase. Our reward definition is based on the Intersection-over-Union (IoU) metric computed between b_t and the ground-truth bounding-box $b_t^{(g)}$, i.e.,

$$\text{IoU}(b_t, b_t^{(g)}) = (b_t \cap b_t^{(g)})/(b_t \cup b_t^{(g)}) \in [0, 1]. \tag{3.2}$$

At every interaction step t, the reward is formally defined as

$$r_t = r(b_t, b_t^{(g)}) = \begin{cases} \omega\left(\text{IoU}(b_t, b_t^{(g)})\right) & \text{if } \text{IoU}(b_t, b_t^{(g)}) \geq 0.5 \\ -1 & \text{otherwise} \end{cases} \tag{3.3}$$

where the function $\omega(z) = 2(\lfloor z \rceil_{0.05}) - 1$ floors to the closest 0.05 digit and shifts the input range from $[0, 1]$ to $[-1, 1]$.

3.2.3 Learning Tracking from Teachers

The student $\mathbf{s}$ is first trained in an offline stage. Through KD, the knowledge is transferred and compressed from a set of $\mathbf{t}$ to $\mathbf{s}$. By means of RL, the ability of evaluating such a knowledge is acquired as well as its enhancement is performed. All the knowledge gained during this learning step will be later used for online tracking.

Hence, the student $\mathbf{s}$ is formally treated as a function $\mathbf{s}(s_t|\theta) : \mathcal{S} \to \mathcal{A} \times \mathbb{R}$ parameterized by the weights θ. Given the state s_t, $\mathbf{s}$ outputs both the action a_t and the state-value v_t. The latter is the prediction of the cumulative reward the student expects to receive from s_t to the end of the interaction. Since the proposed reward definition is a direct measure of the IoU occurring between the predicted and the ground-truth bounding-boxes, v_t gives an estimate of the total amount of IoU that the student expects to obtain from state s_t onwards. Similarly as done for the student, we treat the set of tracking teachers as $\mathbf{T} = \left\{\mathbf{t} : \mathcal{I} \to \mathbb{R}^4\right\}$ where each $\mathbf{t}$ is a function that, given a frame, produces a bounding-box estimate for that frame, i.e. $b_t^{(\mathbf{t})} = \mathbf{t}(F_t)$.[1]

Learning Strategy. To optimize the learnable parameters θ of the student, we propose a single offline end-to-end learning stage (depicted in Figure 3.2) that follows the recent RL trends of using distributed algorithms to speedup the training phase [69, 118]. We distribute an S number of parallel and independent students that share the weights θ. At the beginning of an episode E_j, each student makes a copy θ' of θ. Such θ' are used to interact with M_j to obtain experience that is later exploited in a loss function formulation. The gradients $\nabla_{\theta'}$ of such a loss with respect to θ' are used to asynchronously update the shared set of weights θ. The entire procedure is repeated until the performance of $\mathbf{s}$ on MDPs defined over the videos of a validation set reaches a maximum. To include both the KD and RL objectives in this learning architecture, we devote half of the parallel students, which we refer to as KD students, for the learning of the teachers' tracking policy. The other half, i.e. the RL students, learn to evaluate and improve the distilled policy by interacting with M_j autonomously. Details about the interaction and the losses for the two types of learning students are given in the following. For better clarity, we present the description by referring to a single student.

KD Student. Each KD student interacts with M_j by observing states, performing actions and receiving rewards. To distill knowledge independently from the teachers' inner implementation, the student learns from the actions of the teachers $\mathbf{t} \in \mathbf{T}$ which

[1] At the first frame of a video teachers are initialized with the ground-truth bounding-box enclosing the target.

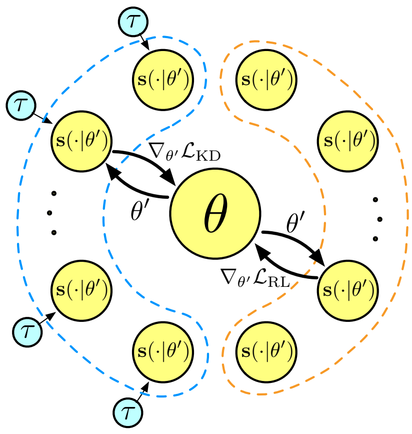

$\mathcal{T}$
$\mathbf{s}(\cdot|\theta')$
$\nabla_{\theta'}\mathcal{L}_{\text{KD}}$
θ'
θ
θ'
$\nabla_{\theta'}\mathcal{L}_{\text{RL}}$

Figure 3.2: Scheme of the proposed KD-RL-based learning framework. S students interact independently with M_j using a copy θ' of the shared weights θ which is updated after every episode. Every t_{max} steps of interaction each student sends the computed gradients to a central process that applies an update on θ. The KD students (highlighted by the blue dashed contour) extract knowledge from teachers $\mathbf{t} \in \mathbf{T}$ by optimizing $\mathcal{L}_{\text{dist}}$. RL students (within the orange dashed contour) learn an autonomous tracking policy by optimizing $\mathcal{L}_{\text{RL}}$.

are executed in parallel. In particular, a teacher $\mathbf{t}$ is exploited every t_{max} steps with the following loss function

$$\mathcal{L}_{\text{dist}} = \sum_{i=1}^{t_{max}} |a_i^{(\mathbf{t})} - \mathbf{s}_\pi(s_i|\theta')| \cdot m_i, \tag{3.4}$$

which is the L1 loss between the actions performed by the student and the actions $a_t^{(\mathbf{t})} = \phi(b_t^{(\mathbf{t})}, b_{t-1})$ that the teacher would take to move the student's bounding-box b_{t-1} into the teacher's prediction $b_t^{(\mathbf{t})}$. The function $\phi : \mathbb{R}^4 \times \mathbb{R}^4 \to \mathcal{A}$ is used to obtain an action given two consecutive bounding-box predictions, and it is defined as

$$\phi(b_t^{(\mathbf{t})}, b_{t-1}^{(\mathbf{t})}) = \begin{cases} \Delta x_t^{(\mathbf{t})} = (x_t^{(\mathbf{t})} - x_{t-1}^{(\mathbf{t})})/w_{t-1}^{(\mathbf{t})} \\ \Delta y_t^{(\mathbf{t})} = (y_t^{(\mathbf{t})} - y_{t-1}^{(\mathbf{t})})/h_{t-1}^{(\mathbf{t})} \\ \Delta w_t^{(\mathbf{t})} = (w_t^{(\mathbf{t})} - w_{t-1}^{(\mathbf{t})})/w_{t-1}^{(\mathbf{t})} \\ \Delta h_t^{(\mathbf{t})} = (h_t^{(\mathbf{t})} - h_{t-1}^{(\mathbf{t})})/h_{t-1}^{(\mathbf{t})}. \end{cases} \tag{3.5}$$

At every time step t, the teacher $\mathbf{t}$ is selected as

$$\mathbf{t} \in \mathbf{T} \; : \; \text{IoU}(b_t^{(\mathbf{t})}, b_t^{(g)}) = \max_{\mathbf{t} \in \mathbf{T}} \text{IoU}(b_t^{(\mathbf{t})}, b_t^{(g)}) \tag{3.6}$$

since we want to always learn from the best teacher. The absolute values of Eq. (3.4) are masked by the values $m_i \in \{0, 1\}$. Each of these is computed along the interaction with M_j and denotes if the student performed worse ($m_i = 1$) or better ($m_i = 0$) than the

Algorithm 1 Pseudo-code for each parallel KD student.

```
// Assume global shared parameters $\theta$
// Assume parallel student parameters $\theta'$
Set $j \leftarrow 0$
Initialize $M_j$ for video $\mathcal{V}_j$
Set $t \leftarrow 0$
repeat
    Reset gradients: $d\theta \leftarrow 0$
    Synchronize student parameters $\theta' = \theta$
    Set $t_{start} \leftarrow t$
    repeat
        Get state $s_t$
        Perform action $a_t$ according to $\mathbf{s}_\pi(s_t \mid \theta')$
        Run the best $\mathbf{t}$ with $F_t$ and obtain $b_t^{(\mathbf{t})}$
        Obtain teacher action $a_t^{(\mathbf{t})} = \phi(b_t^{(\mathbf{t})}, b_t)$
        Receive reward $r_t$, teacher reward $r_t^{(\mathbf{t})}$ and
          new state $s_{t+1}$
        if $r_t \geq r_t^{(\mathbf{t})}$ then
            $m_t \leftarrow 0$
        else
            $m_t \leftarrow 1$
        end if
        $t \leftarrow t + 1$
    until terminal state $s_{\widehat{T}_j}$ or $t - t_{start} = t_{max}$
    Reset loss: $\mathcal{L}_{\text{dist}} \leftarrow 0$
    for $i \in \{t_{start}, \cdots, t\}$ do
        $\mathcal{L}_{\text{dist}} \leftarrow \mathcal{L}_{\text{dist}} + |a_i^{(\mathbf{t})} - a_i| \cdot m_i$
    end for
    Compute gradients w.r.t. $\theta'$ $d\theta \leftarrow \nabla_{\theta'} \mathcal{L}_{\text{dist}}$
    Perform asynchronous update of $\theta$ using $d\theta$
    if terminal state $s_T$ is reached then
        Set $j \leftarrow (j + 1) \mod |\mathcal{D}|$
        Initialize $M_j$ for video $\mathcal{V}_j$
        $t \leftarrow 0$
    end if
until stopping criteria are met
```

teacher in terms of the respective rewards $r(s_t, a_t)$ and $r(s_t, a_t^{(\mathbf{t})})$. Eq. (3.4) is similar to what [112] introduced for KD from bounding-box predictions of object detectors. Differently, here we provide a temporal formulation of such an objective and we swap the L2 with the L1 loss, which was shown to work better for optimizing regression-based trackers [41, 42]. The pseudocode of the procedure followed by each KD student is presented in Algorithm 1.

In simple terms, the optimization of the loss defined in Eq. (3.4) changes the weights θ only if the student's performance is lower than the performance of the teacher. In this way, we make the teacher transfer its knowledge only when the student performs bad. In the others, we let the student free to follow its current tracking policy since it is superior.

RL Student. The learning process performed by an RL student is similar to the standard RL procedure for continuous control [55]. At each step t of a t_{max}-long interaction, the student predicts the expected cumulative reward $v_t = \mathbf{s}_v(s_t|\theta')$. At the same time, it samples a new action from a normal distribution $\mathcal{N}(\mu, \sigma)$ where the mean is defined as the student's predicted action, $\mu = \mathbf{s}_\pi(s_t|\theta')$, and the standard deviation is obtained as $\sigma = |\mathbf{s}_\pi(s_t|\theta') - \phi(b_t^{(g)}, b_{t-1})|$. The latter is the absolute distance between the student's action and the action that should be taken to reach the ground-truth bounding-box $b_t^{(g)}$ from b_{t-1}. Intuitively, the distribution $\mathcal{N}$ shrinks when a_t is close to the ground-truth action $\phi(b_t^{(g)}, b_{t-1})$, thus reducing the chance of choosing potential wrong actions when approaching the correct one. On the other hand, when a_t is far from $\phi(b_t^{(g)}, b_{t-1})$, $\mathcal{N}$ spreads forcing the student to explore more. The just described exploration procedure is performed to improve the learning of the state-value function and of the distilled actions. Indeed, trying new actions allows to reach new states that in turn generate different cumulative reward values. In addition, it allows to discover actions that are different from those of the teachers' tracking behaviors, and to optimize them directly for the bounding-box overlap expressed through r_t.

After the end of the interaction, the state-value predictions are optimized by the following value loss formulation

$$\mathcal{L}_v = \sum_{i=1}^{t_{max}} \frac{1}{2}\big(R_i - \mathbf{s}_v(s_i|\theta')\big)^2, R_i = \sum_{k=1}^{i} \gamma^{k-1} r_k, \tag{3.7}$$

while the sampled actions are optimized via the policy loss

$$\mathcal{L}_\pi = -\sum_{i=1}^{t_{max}} \log \mathbf{s}_\pi(s_i|\theta')\big(r_i + \gamma \mathbf{s}_v(s_{i+1}|\theta') - \mathbf{s}_v(s_i|\theta')\big). \tag{3.8}$$

Overall, the two aforementioned losses are combined in the following definition

$$\mathcal{L}_{\mathrm{RL}} = \mathcal{L}_v + \mathcal{L}_\pi \tag{3.9}$$

which consists in the standard advantage actor-critic objective formulation [58]. For more details, the learning procedure of the RL students is given as pseudocode in Algorithm 2.

Curriculum Learning. A curriculum learning strategy [89] is designed to further facilitate and improve the student's learning. At the beginning of the learning process, the length of each episode E_j is set to $\widehat{T}_j = 1$. After terminating each E_j, a success counter C_j for the MDP M_j is increased if the student performs better than the teacher for that interaction, i.e. if the following condition holds

$$\sum_{i=1}^{\widehat{T}_j} r_i \geq \sum_{i=1}^{\widehat{T}_j} r_i^{(\mathbf{t})}. \tag{3.10}$$

The episode's termination index $1 \geq \widehat{T}_j \geq T_j$ is successively increased during the training procedure by checking if $\frac{C_j}{E_j} \geq \lambda$. After each update of $\widehat{T}_j$, C_j is reset to zero.

With this setup, we ensure that for every increase of $\widehat{T}_j$ the student faces a simpler learning problem where a single-time-step action needs to be optimized. Hence, the student is likely to succeed in a shorter time, since it has already developed a tracking policy that, up to $\widehat{T}_j - 1$, is at least good as the one of the teachers.

Algorithm 2 Pseudo-code for each parallel RL student.

```
// Assume global shared parameters $\theta$
// Assume parallel student parameters $\theta'$
Set $j \leftarrow 0$
Initialize $M_j$ for video $\mathcal{V}_j$
Set $t \leftarrow 0$
repeat
    Reset gradients: $d\theta \leftarrow 0$
    Synchronize student parameters $\theta' = \theta$
    Set $t_{start} \leftarrow t$
    repeat
        Get state $s_t$
        Sample action $a_t$ according to $\mathcal{N}(\mathbf{s}_\pi(s_t|\theta'), |\mathbf{s}_\pi(s_t|\theta') - \phi(b_t^{(g)}, b_{t-1})|)$
        Receive reward $r_t$ and new state $s_{t+1}$
        $t \leftarrow t + 1$
    until terminal $s_{\widehat{T_j}}$ or $t - t_{start} = t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_{\widehat{T_j}} \\ \mathbf{s}_v(s_t|\theta') & \text{for non-terminal } s_t \end{cases}$
    Reset losses: $\mathcal{L}_v \leftarrow 0, \mathcal{L}_\pi \leftarrow 0$
    for $i \in \{t-1, \cdots, t_{start}\}$ do
        $R \leftarrow r_i + \gamma R$
        $\mathcal{L}_v \leftarrow \mathcal{L}_v + \frac{1}{2}\left(R - \mathbf{s}_v(s_i|\theta')\right)^2$
        $\mathcal{L}_\pi \leftarrow \mathcal{L}_\pi - \log \mathbf{s}_\pi(s_i|\theta')(r_i + \gamma \mathbf{s}_v(s_{i+1}|\theta') - \mathbf{s}_v(s_i|\theta'))$
    end for
    $\mathcal{L}_{\text{RL}} \leftarrow \mathcal{L}_v + \mathcal{L}_\pi$
    Compute gradients w.r.t. $\theta'$ $d\theta \leftarrow \nabla_{\theta'} \mathcal{L}_{\text{RL}}$
    Perform asynchronous update of $\theta$ using $d\theta$
    if terminal state $s_T$ is reached then
        Set $j \leftarrow (j+1) \mod |\mathcal{D}|$
        Initialize $M_j$ for video $\mathcal{V}_j$
        $t \leftarrow 0$
    end if
until stopping criteria are met
```

3.2.4 Student Architecture

The architecture used to maintain the representation of the student, which is depicted in Figure 3.3, presents a structure similar to the deep regression trackers of [41, 42]. The choice of employing such a design is motivated by the fact that those trackers are established methods for their simplicity and high processing speed. Hence, we think they are a good fit to represent a compressed student model. Moreover, their input/output interfaces well integrate with the state and action definitions of the proposed MDP.

Therefore, our student network receives as input the two image patches of s_t which pass through two convolutional branches that have the form of a ResNet-18 architecture [103]. The feature maps produced by the branches are linearized, concatenated together and fed to two consecutive fully connected layers with ReLU activations. After that, the features are given to an LSTM [87] layer. Both the fully connected layers and the LSTM are composed of 512 neurons. The output of the LSTM is finally fed to two separate fully connected heads, one that outputs the action $a_t = \mathbf{s}_\pi(s_t|\theta)$ and the other that outputs the state-value $v_t = \mathbf{s}_v(s_t|\theta)$.

3.2.5 Online Tracking after Learning

After the learning process is completed, the student $\mathbf{s}$ is ready to be used for tracking. Particularly, the capabilities of the student model can be exploited interchangeably

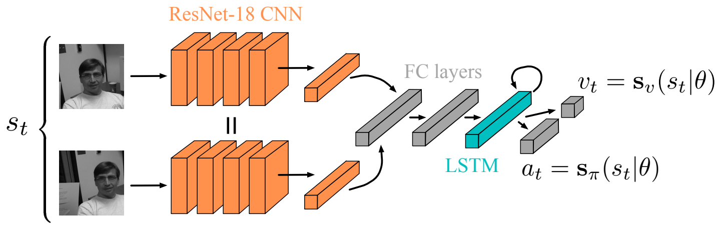

ResNet-18 CNN
FC layers
LSTM
s_t
$v_t = \mathbf{s}_v(s_t|\theta)$
$a_t = \mathbf{s}_\pi(s_t|\theta)$

Figure 3.3: The student architecture is composed by two branches of convolutional layers with shared weights (orange boxes) followed by two fully-connected layers (gray boxes), an LSTM layer (in teal), and two parallel fully connected layers for the prediction of a_t and v_t respectively.

without any further adaptation in the following three modalities, where:

(i) the learned state-value function $\mathbf{s}_v(s_t|\theta)$ is used to evaluate the performance of the pool of teachers $\mathbf{T}$ in order to choose the best $b_t^{(\mathbf{t})}$ for target localization and hence perform tracker fusion. We call this setup TRASFUST (TRAcking by Student FUSing Teachers).

(ii) the learned policy $\mathbf{s}_\pi(s_t|\theta)$ and value function $\mathbf{s}_v(s_t|\theta)$ are used to, respectively, predict b_t and evaluate $\mathbf{s}$ and $\mathbf{t}$ tracking behaviors, in order to correct the $\mathbf{s}$'s performance with $\mathbf{t}$. We refer to this setup as TRAST (TRAcking Student and Teacher).

(iii) the student's learned policy $\mathbf{s}_\pi(s_t|\theta)$ is used to predict the bounding-boxes b_t independently from the teachers. We call this setting TRAS (TRAcking Student).

In the following we provide precise explanations about the three setups. For all the settings, each tracking video $\mathcal{V}$, with target object outlined by $b_0^{(g)}$ in F_0, is considered as the MDP described in section 3.2.2.

(i) TRASFUST. In this tracking setup, the student's learned state-value function $\mathbf{s}_v(s_t|\theta)$ and the pool of teachers $\mathbf{T}$ are exploited. At each step t, teachers $\mathbf{t} \in \mathbf{T}$ are executed following their standard methodology. States $s_t^{(\mathbf{t})} = \rho(F_{t-1}, F_t, b_{t-1}^{(\mathbf{t})}, c)\ \forall \mathbf{t} \in \mathbf{T}$ are obtained. Then, the expected future performance $v_t^{(\mathbf{t})} = \mathbf{s}_v(s_t^{(\mathbf{t})}|\theta)$ is computed through the student for each of the teachers. Based on such values, the output bounding-box b_t for the target in the current frame is selected among the teachers' predictions $b^{(\mathbf{t}')}$ by considering the teacher $\mathbf{t}'$ that achieves the highest expected return, i.e.

$$\mathbf{t}' \in \mathbf{T} \ : \ v_t^{(\mathbf{t}')} = \max_{\mathbf{t} \in \mathbf{T}} v_t^{(\mathbf{t})}. \tag{3.11}$$

Overall, this procedure consists in fusing sequence-wise the predictions of the teacher set $\mathbf{T}$. Notice that, at every t, the execution of each $\mathbf{t} \in \mathbf{T}$ is independent from the others' and from the execution of $\mathbf{s}$. Indeed, the student does not need to wait for the teachers to finish their processing on F_t to evaluate them. This is possible because $v_t^{(\mathbf{t})}$

are obtained after $s_t^{(\mathbf{t})}$ which are computed using the bounding-box predictions $b_{t-1}^{(\mathbf{t})}$ given at the previous frame. Hence, with this setting, the processing of teachers and student can be executed in parallel to improve efficiency.

(ii) TRAST. In this setup, the student makes use of the learned $\mathbf{s}_\pi(s_t|\theta)$ to predict b_t and of $\mathbf{s}_v(s_t|\theta)$ to evaluate its tracking quality and the one of a $\mathbf{t}$ which is run in parallel. In particular, at each time step t, $v_t = \mathbf{s}_v(s_t|\theta)$ and $v_t^{(\mathbf{t})} = \mathbf{s}_v(s_t^{(\mathbf{t})}|\theta)$ are obtained as performance evaluations for $\mathbf{s}$ and $\mathbf{t}$ respectively, where the teacher state is computed as $s_t^{(\mathbf{t})} = \rho(F_{t-1}, F_t, b_{t-1}^{(\mathbf{t})}, c)$. Then, after the comparison of the two state-values, TRAST decides if to localize the target with the student's or the teacher's predicted bounding-box. More formally, if $v_t \geq v_t^{(\mathbf{t})}$ then $b_t := \psi(a_t, b_{t-1})$ otherwise $b_t := b_t^{(\mathbf{t})}$. Notice that the second assignment has the side effect of correcting the tracking behavior of the student. Indeed, in such a scenario, at step $t+1$ the previously known bounding-box from which the new state s_{t+1} is computed matches the teacher's prediction. Overall, the TRAST modality consists in an online adaption procedure that evaluates $\mathbf{t}$'s performance based on its bounding-box predictions and that eventually passes control to it. As for TRASFUST, the execution of teacher and student can be run in parallel because they do not depend on each other, and the evaluation of the teacher requires just the prediction $b_{t-1}^{(\mathbf{t})}$ available at the previous time step.

(iii) TRAS. In this final modality, just the student's learned actions $\mathbf{s}_\pi(s_t|\theta)$, which represent the compressed tracking knowledge of the teachers, are exploited. At each step t, the state s_t is extracted from frames F_{t-1}, F_t, the action $a_t = \mathbf{s}_\pi(s_t|\theta)$ is obtained and transformed into the target box $b_t = \psi(a_t, b_{t-1})$. This setup is computationally light and fast as it requires just a forward pass through the student's network to obtain a localization for the target.

3.3 Experimental Setup

3.3.1 Teachers

The tracking teachers picked for this study were KCF [32], MDNet [40], ECO [60], SiamRPN [44], ATOM [61], and DiMP [62]. The selection was motivated by the fact that these trackers represent the most popular approaches in the visual tracking panorama. Moreover, they can provide different complementary knowledge since they tackle visual tracking by different strategies. In the experiments, we considered exploiting a single teacher or a pool of teachers. In particular, the following sets of teachers were examined $\mathbf{T}_{\texttt{K}} = \{\texttt{KCF}\}, \mathbf{T}_{\texttt{M}} = \{\texttt{MDNet}\}, \mathbf{T}_{\texttt{E}} = \{\texttt{ECO}\}, \mathbf{T}_{\texttt{S}} = \{\texttt{SiamRPN++}\}, \mathbf{T}_{\texttt{A}} = \{\texttt{ATOM}\}, \mathbf{T}_{\texttt{D}} = \{\texttt{DiMP}\}, \mathbf{T}_{\texttt{P}} = \{\texttt{KCF}, \texttt{MDNet}, \texttt{ECO}, \texttt{SiamRPN++}\}$.

3.3.2 Transfer Set

Experiments were performed on two transfer sets. The first is the training set of GOT-10k dataset [39], which provides 9335 videos. The second is the training set of the LaSOT benchmark [38] which contains 1120 videos. Just the sets of teachers $\mathbf{T}_{\texttt{K}}, \mathbf{T}_{\texttt{M}}, \mathbf{T}_{\texttt{E}}, \mathbf{T}_{\texttt{S}}, \mathbf{T}_{\texttt{P}}$

Table 3.1: Teacher-based statistics of the GOT-10k-based transfer set. The number of trajectories, average overlap (AO), and size of the transfer set $|\mathcal{D}|$ are reported for every teacher set and for every action bounding-box threshold β.

Teachers	$\beta = 0.5$			$\beta = 0.6$			$\beta = 0.7$			$\beta = 0.8$			$\beta = 0.9$		
	# traj	AO	$\|\mathcal{D}\|$	# traj	AO	$\|\mathcal{D}\|$	# traj	AO	$\|\mathcal{D}\|$	# traj	AO	$\|\mathcal{D}\|$	# traj	AO	$\|\mathcal{D}\|$
$\mathbf{T}_{\mathtt{K}}$	1884	0.798	9225	1097	0.836	5349	439	0.873	2122	73	0.914	356	0	0.0	0
$\mathbf{T}_{\mathtt{M}}$	1600	0.767	7859	781	0.808	3831	216	0.851	1052	18	0.898	86	0	0.0	0
$\mathbf{T}_{\mathtt{E}}$	2754	0.808	13526	1659	0.843	8122	720	0.879	3507	160	0.915	773	1	0.954	4
$\mathbf{T}_{\mathtt{S}}$	3913	0.829	19259	2646	0.854	12997	1447	0.878	7080	431	0.908	2097	9	0.947	42
$\mathbf{T}_{\mathtt{P}}$	4519	0.840	22252	3092	0.863	15195	1698	0.887	8307	496	0.915	2414	10	0.948	46

Table 3.2: Teacher-based statistics of the LaSOT transfer set. The number of trajectories, success score (SS) [71], and size of the transfer set $|\mathcal{D}|$ are reported for every teacher set and for action quality threshold $\beta = 0.5$.

Teachers	$\beta = 0.5$		
	# traj	SS	$\|\mathcal{D}\|$
$\mathbf{T}_{\mathtt{K}}$	16	0.835	80
$\mathbf{T}_{\mathtt{M}}$	32	0.830	160
$\mathbf{T}_{\mathtt{E}}$	44	0.817	220
$\mathbf{T}_{\mathtt{S}}$	87	0.852	435
$\mathbf{T}_{\mathtt{P}}$	106	0.856	530

were used for offline learning, as none of these was trained on these datasets ($\mathbf{T}_{\mathtt{A}}$ and $\mathbf{T}_{\mathtt{D}}$ are trained with data extracted from the GOT-10k and LaSOT training sets). This is an important point because unbiased examples of the trackers' behaviour should be exploited to train the student. Moreover, predictions that exhibit meaningful knowledge should be retained. Therefore, for each teacher, we filtered out all the videos $\mathcal{V}_j$ in which the predictions did not satisfy $\mathrm{IoU}(b_t^{(\mathbf{t})}, b_t^{(g)}) > \beta$ for all $t \in \{1, \ldots, T_j\}$. We considered $\beta = 0.5$ as minimum threshold for a prediction to be considered positive and, for the GOT-10k dataset, we then varied β among 0.6, 0.7, 0.8, 0.9 to evaluate the student's performance under different conditions of teachers' predictions quality and data quantity. For the LaSOT transfer set, we used $\beta = 0.5$ as this value resulted in a low number of usable sequences. To produce more training samples, we followed a similar procedure to [42] and splitted each video (and the respective teacher predictions and ground-truth bounding-box sequences) in five sequences of 32 frames with random starting index. Table 3.1 presents a summary of the GOT-10k-based $\mathcal{D}$. The number of positive trajectories (i.e. video-long sequences of teacher predictions), the average overlap (AO) [39] on the transfer set, and the total number of sequences $|\mathcal{D}|$ after the split in chunks of 32 frames, are reported per teacher and per β. Similar statistics for the LaSOT-based $\mathcal{D}$ are given in Table 3.2.

3.3.3 Benchmarks and Performance Measures

In this subsection, we provide the details about the benchmark datasets and the performance measures we employed to validate our solution.

GOT-10k Test Set. The GOT-10k test set [39] is composed of 180 videos where target objects belong to 84 different classes and 32 forms of object motion are present.

Table 3.3: Performance of the proposed tracking modalities in comparison with the teachers. Best tracker, per benchmark and measure, is highlighted in red, second-best in blue.

	GOT-10k			UAV123		LaSOT		OTB-100		FPS
	AO	$SR_{0.50}$	$SR_{0.75}$	SS	PS	SS	PS	SS	PS	
TRASFUST	**0.617**	**0.729**	**0.490**	**0.679**	**0.873**	**0.576**	**0.574**	**0.701**	**0.931**	20
TRAST	0.604	0.708	0.469	0.647	0.837	0.545	0.524	0.643	0.865	25
TRAS	0.484	0.556	0.326	0.515	0.655	0.386	0.330	0.481	0.644	90
KCF [32]	0.203	0.177	0.065	0.331	0.503	0.178	0.166	0.477	0.693	105
MDNet [40]	0.299	0.303	0.099	0.489	0.718	0.397	0.373	0.673	**0.909**	5
ECO [60]	0.316	0.309	0.111	0.532	0.726	0.324	0.301	0.668	0.896	15
SiamRPN [44]	0.508	0.604	0.308	0.616	0.785	0.508	0.492	0.649	0.851	43
ATOM [61]	0.556	0.634	0.402	0.643	0.832	0.516	0.506	0.660	0.867	20
DiMP [62]	**0.611**	**0.717**	**0.492**	**0.653**	**0.839**	**0.570**	**0.569**	**0.681**	0.888	25

The evaluation protocol employed is the one-pass evaluation (OPE) [71], while the metrics used are the average overlap (AO) and the success rates (SR) with overlap thresholds 0.50 ($SR_{0.50}$) and 0.75 ($SR_{0.75}$).

OTB-100. The OTB-100 benchmark [71] is a set of 100 challenging videos and it is widely used in the tracking literature. The standard evaluation procedure for this dataset is the OPE method, while the Area Under the Curve (AUC) of the success and precision plots, referred as success score (SS) and precision scores (PS) respectively, are utilized to quantify the performance of the trackers.

UAV123. The UAV123 benchmark [70] presents 123 videos that are inherently different from traditional benchmarks like OTB and VOT [119], since it offers sequences acquired from low-altitude UAVs. To evaluate the trackers, the standard OTB methodology [71] is employed.

LaSOT. A performance evaluation was also conducted on the test set of LaSOT benchmark [38]. This dataset is composed of 280 videos with a total of more than 650k frames and an average sequence length of 2500 frames. The latter statistic is much higher than the same for the aforementioned benchmarks. The same evaluation methodology and metrics used for the OTB experiments [71] are employed for this dataset.

VOT2020. The VOT benchmarks are datasets used in the annual VOT tracking competitions. These sets change year by year introducing increasingly challenging tracking scenarios. We evaluated our trackers on the set of the VOT2020 challenge [119], which provides 60 highly challenging videos. Within the framework used by the VOT committee, trackers are evaluated with the Expected Average Overlap (EAO), Accuracy (A), and Robustness (R) [119] based on target segmentation masks. Differently from the OPE, the VOT evaluation protocol presents the automatic re-initialization of the tracker at multiple initialization points defined along a video.

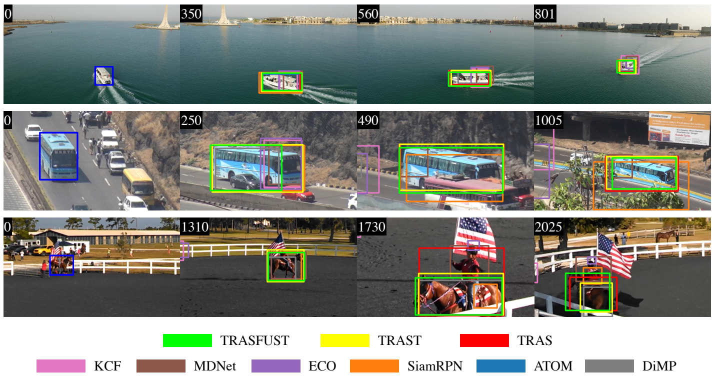


Figure 3.4: Qualitative examples of the proposed tracking modalities TRASFUST, TRAST, TRAS, in comparison with trackers used as teachers.

3.3.4 Implementation Details

The image crops composing states s_t were resized to $[128 \times 128 \times 3]$ pixels and standardized by the mean and standard deviation calculated on the ImageNet dataset [19]. The ResNet-18 weights of the student model were pre-trained for image classification on the same dataset [19]. The image context factor c was set to 1.5. The training videos were processed in chunks of 32 frames as [42]. Due to hardware constraints, a maximum of $S = 24$ students were distributed for training on 4 NVIDIA TITAN V GPUs of a machine with an Intel Xeon E5-2690 v4 @ 2.60GHz CPU and 320 GB of RAM. The discount factor γ was set to 1. The length of the interaction before an update was defined in $t_{max} = 5$ steps. The curriculum learning parameter λ was set to 0.25. The Radam optimizer [120] was employed and the learning rate for both KD and RL students was set to 10^{-6}. A weight decay of 10^{-4} was also added to $\mathcal{L}_{\text{dist}}$ as regularization term. To control the magnitude of the gradients and stabilize learning, $\mathcal{L}_{\text{RL}}$ was multiplied by 10^{-3}. The student was trained until the validation performance on the videos of the GOT-10k validation set stopped improving. Longest trainings took around 10 days. At test time, following [42], the LSTM's hidden state is reset every 32 frames to the one hidden state computed after the first student prediction (i.e. $t = 1$) to maintain reference to the target. The speed of the parallel setups of TRASFUST and TRAST was computed by considering the speed of the slowest tracker (student or teacher) plus an overhead. Code was implemented in Python. The source code publicly available was used to implement the teacher trackers. Default configurations were respected. For fair comparison, we report the results of such implementations, that have slightly different performance than those stated in the original papers. For the comparison with other state-of-the-art solutions, we report the results presented in the original papers.

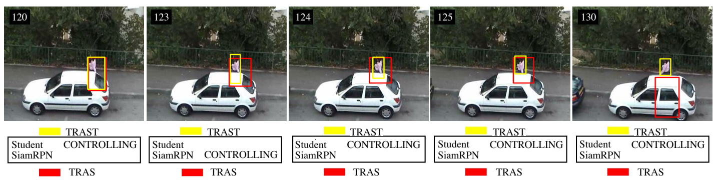


Figure 3.5: Visual example of how TRAST relies effectively on the teacher, passing control to $\mathbf{T}_{\mathrm{S}}$ and saving the simple student tracking policy (TRAS) from a drift.

3.4 Results

3.4.1 General Remarks

Table 3.3 reports the performances of the best configurations of TRASFUST, TRAST, and TRAS in comparison with the teacher trackers. The performance of TRASFUST demonstrates the effectiveness of the proposed tracker fusion strategy based on the student's evaluation ability. In fact, this setup results in the most accurate and robust tracking modality in general. Benefiting a teacher for student correction during tracking is a valid online adaptation procedure. This modality achieves lower performance than TRASFUST, but requires less computational resources as just a single teacher is executed. A qualitative example of TRAST's ability to pass control to the teacher is given in Figure 3.5. Finally, TRAS is the lightest and fastest modality, showing a good accuracy with an high processing speed of 90 FPS. Although in general student models cannot outperform their teachers due to their simple and compressed architecture [121], TRAST and TRAS exhibit such behavior on benchmarks where teachers are weak. It is worth noticing that all the three modalities show balanced performance across the benchmarks, thus demonstrating good generalization. In Figure 3.4 some qualitative examples of the proposed modalities are presented in comparison to the teachers. Overall, the results achieved demonstrate that the proposed student-teacher tracker respects, respectively, the goals (i), (ii), (iii) introduced in Section 3.

3.4.2 Analysis

In this section, we analyze in depth the capabilities of the proposed tracking solution. If not specified otherwise, the three tracking setups exploit the student trained using $\mathbf{T}_{\mathrm{P}}$ and $\beta = 0.5$, while the default TRASFUST uses the teacher set $\mathbf{T}_{\mathrm{P}}$, and the default TRAST exploits the teacher $\mathbf{T}_{\mathrm{S}}$.

Tracking Ability. Table 3.4 shows the performance of TRASFUST and TRAST with different configurations of the teacher set used for tracking. The fusion mechanism of TRASFUST performs at its best with two teachers. Whenever weaker teachers are added to the pool, the performance tends to decrease, suggesting a behavior similar to the one pointed out in [100]. Figure 3.6 reports the fractions of teacher predictions that form the TRASFUST's outputs. It can be noted that they reflect the distribution of the teachers' performance on the original transfer set (shown in Table 3.1). To make sure that the decisions taken by TRASFUST are meaningful, we analyzed some baseline fu-

Table 3.4: Performance of TRASFUST and TRAST while considering different teacher setups for tracking. Best results per tracker are highlighted in bold.

	Teachers	GOT-10k			UAV123		LaSOT		OTB-100		FPS
		AO	$\mathrm{SR}_{0.50}$	$\mathrm{SR}_{0.75}$	SS	PS	SS	PS	SS	PS	
TRASFUST	$\mathbf{T}_\mathtt{K} \cup \mathbf{T}_\mathtt{M}$	0.317	0.319	0.105	0.493	0.720	0.396	0.372	0.666	0.901	5
	$\mathbf{T}_\mathtt{M} \cup \mathbf{T}_\mathtt{E}$	0.384	0.398	0.131	0.563	0.791	0.422	0.392	**0.701**	**0.931**	5
	$\mathbf{T}_\mathtt{E} \cup \mathbf{T}_\mathtt{S}$	0.526	0.624	0.305	0.634	0.815	0.507	0.500	0.670	0.877	15
	$\mathbf{T}_\mathtt{A} \cup \mathbf{T}_\mathtt{D}$	**0.617**	**0.729**	**0.490**	**0.679**	**0.873**	**0.576**	**0.574**	0.692	0.895	20
	$\mathbf{T}_\mathtt{M} \cup \mathbf{T}_\mathtt{E} \cup \mathbf{T}_\mathtt{S}$	0.517	0.615	0.294	0.633	0.823	0.513	0.504	0.682	0.897	5
	$\mathbf{T}_\mathtt{P}$	0.519	0.616	0.287	0.628	0.823	0.510	0.505	0.675	0.899	5
TRAST	$\mathbf{T}_\mathtt{K}$	0.469	0.541	0.297	0.562	0.727	0.422	0.376	0.560	0.760	90
	$\mathbf{T}_\mathtt{M}$	0.494	0.573	0.302	0.604	0.798	0.466	0.431	0.596	0.815	5
	$\mathbf{T}_\mathtt{E}$	0.521	0.607	0.307	0.606	0.795	0.456	0.419	0.608	0.822	15
	$\mathbf{T}_\mathtt{S}$	0.531	0.626	0.345	0.603	0.773	0.490	0.470	0.604	0.818	40
	$\mathbf{T}_\mathtt{A}$	0.557	0.640	0.393	0.634	0.823	0.513	0.488	0.623	0.838	20
	$\mathbf{T}_\mathtt{D}$	**0.604**	**0.708**	**0.469**	**0.647**	**0.837**	**0.545**	**0.524**	**0.643**	**0.865**	25

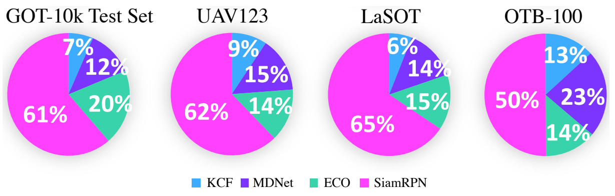


Figure 3.6: Per benchmark pie plots showing the impact of each of teacher on the fusion strategy of TRASFUST.

sion methods for different pools of teachers. In particular, we considered fusion methods where at each time step t: the $b_t^{(\mathbf{t})}$ having maximum $\mathrm{IoU}(b_t^{(\mathbf{t})}, b_t^{(g)})$ is selected (max); the $b_t^{(\mathbf{t})}$ having worst $\mathrm{IoU}(b_t^{(\mathbf{t})}, b_t^{(g)})$ is selected (min); the mean of the teachers' bounding-box predictions is computed (mean); a random bounding-box prediction is chosen among the ones produced by the teachers (random). Notice that, at test-time, the max and min setups need an oracle to compute the ground-truth bounding-box $b_t^{(g)}$. The results of these baselines are reported in Table 3.5 in comparison with TRASFUST. The latter outperforms the random and mean fusion method, which are the only achievable strategies without the availability of ground-truth data. Overall, the general performance trends of TRASFUST and TRAST reflect the increasing tracking capabilities of the teachers. Indeed, on every considered benchmark, the performance increases as stronger teachers are employed. This is additionally proven by Figure 3.6 and by Figure 3.7 (a) that show that TRASFUST and TRAST employ more the output of teachers as they perform better. Moreover, the two tracking modalities show to be unbiased to the training teachers, as their capabilities generalize also to $\mathbf{T}_\mathtt{A}$ and $\mathbf{T}_\mathtt{D}$ which were not exploited during training. Moreover, it is worth to mention that part of the error committed by TRASFUST and TRAST on benchmarks like OTB-100 is due to the knowledge on object entities acquired by the student during learning. As demonstrated by Figure 3.8, our proposed tracking modalities localize objects exploiting their entire appearance. In situations of ambiguous ground-truths, this behavior results in qualitatively better but quantitatively worse predictions, ultimately causing an apparent lower overall quantitative performance.

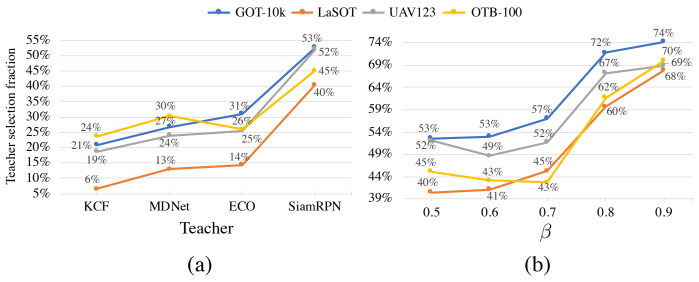


Figure 3.7: (a) Per benchmark fractions of outputs attributed to $\mathbf{t}$ while considering different teachers in the TRAST setup. (b) Per benchmark fractions of outputs attributed to $\mathbf{T}_S$ in TRAST while considering different teacher action thresholds β.

As discussed in the respective paragraphs of the Section 3.2.5, the computation of student and teachers in TRASFUST and TRAST can be parallelized. Hence, the processing speed resulting from this configuration is the lowest one that is achieved by the student or by a teacher in the pool. Since executing the student takes around 90 FPS, TRASFUST and TRAST can run at real-time speed if the teachers in the pool do so.

Student Learning. Here we analyze the tracking performance of our solution considering different settings of the learning strategy of the student.

The results presented in Table 3.6 report the performance of the tracking modalities after the removal of the key components of the proposed learning strategy. No use of the curriculum learning strategy (TRASFUST-no-curr, TRAST-no-curr, TRAS-no-curr) leads to a slightly decreased performance of all the three modalities. Additionally to the inability of exploiting trackers online and thus preventing the development of the TRASFUST and TRAST modalities, considering just the KD objective (Eq. (3.4)) for learning results in lower TRAS performance (TRAS-KD). However, such an optimization strategy works better than learning just from the ground-truth bounding-boxes as proposed in [41, 42] (TRAS-GT), and from both the teachers and the ground-truths as in the original KD framework [102] (TRAS-KD-GT). The results of the student trained only by RL (Eq. (3.9)) are not reported because convergence was not attained. We hypothesize this is due to the large state and action spaces generated by the proposed MDP definition which prevent a safe convergence.

Table 3.7 shows that a good evaluation ability for TRASFUST is achieved even when learning from a single teacher. On the other hand, using more than one teacher during training leads to better action predictions and hence to superior tracking policies that improve the performance of the TRAST and TRAS modalities.

Table 3.8 presents the performance of the proposed modalities while considering different quality of teacher actions. Increasing the quality, thus reducing the number of videos, results in decreasing the accuracy of all the three setups. The loss is not so significant between $\beta = 0.5$ and $\beta = 0.7$, while considering more precise actions TRAS suffers majorly, suggesting that more data is a key factor for an autonomous tracking policy. Interestingly, TRAST and TRASFUST are able to perform tracking even if the

Table 3.5: AO (for the GOT-10k benchmark) and SS (for the others) performance of four baseline teacher fusion methods in comparison with TRASFUST. min reports the results of choosing $b_t^{(\mathbf{t})}$ with minimum overlap with $b_t^{(g)}$. mean reports the result of computing the mean bounding box of the teachers (mean of the coordinates and of width and height). max shows the performance of selecting $b_t^{(\mathbf{t})}$ with highest IoU with $b_t^{(g)}$. random reports the performance of randomly selecting $b_t^{(\mathbf{t})}$.

Teachers	GOT-10k	
	min / mean / max / random	TRASFUST
$\mathbf{T}_{\mathtt{K}} \cup \mathbf{T}_{\mathtt{M}}$	- / 0.222 / - / 0.251	0.317
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}}$	- / 0.266 / - / 0.307	0.384
$\mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	- / 0.355 / - / 0.412	0.526
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	- / 0.311 / - / 0.374	0.517
$\mathbf{T}_{\mathtt{P}}$	- / 0.266 / - / 0.332	0.519
Teachers	UAV123	
	min / mean / max / random	TRASFUST
$\mathbf{T}_{\mathtt{K}} \cup \mathbf{T}_{\mathtt{M}}$	0.313 / 0.376 / 0.511 / 0.411	0.493
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}}$	0.431 / 0.500 / 0.590 / 0.510	0.563
$\mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	0.473 / 0.557 / 0.674 / 0.573	0.634
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	0.407 / 0.521 / 0.692 / 0.546	0.633
$\mathbf{T}_{\mathtt{P}}$	0.294 / 0.419 / 0.694 / 0.492	0.628
Teachers	LaSOT	
	min / mean / max / random	TRASFUST
$\mathbf{T}_{\mathtt{K}} \cup \mathbf{T}_{\mathtt{M}}$	0.162 / 0.235 / 0.414 / 0.288	0.396
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}}$	0.271 / 0.344 / 0.451 / 0.361	0.422
$\mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	0.277 / 0.369 / 0.543 / 0.410	0.507
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	0.242 / 0.371 / 0.579 / 0.406	0.513
$\mathbf{T}_{\mathtt{P}}$	0.140 / 0.288 / 0.584 / 0.349	0.510
Teachers	OTB-100	
	min / mean / max / random	TRASFUST
$\mathbf{T}_{\mathtt{K}} \cup \mathbf{T}_{\mathtt{M}}$	0.474 / 0.551 / 0.710 / 0.590	0.666
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}}$	0.607 / 0.671 / 0.734 / 0.670	0.701
$\mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	0.588 / 0.665 / 0.729 / 0.658	0.670
$\mathbf{T}_{\mathtt{M}} \cup \mathbf{T}_{\mathtt{E}} \cup \mathbf{T}_{\mathtt{S}}$	0.556 / 0.665 / 0.765 / 0.664	0.682
$\mathbf{T}_{\mathtt{P}}$	0.430 / 0.586 / 0.777 / 0.625	0.675

student is trained with limited training samples. Indeed, the plot (b) of Figure 3.7 additionally confirms that the student relies more to its teacher as its tracking policy loses performance. We found thresholds $\beta < 0.5$ to not impact on increased performance, but to lead just to longer training times as more videos are made available.

The performance achieved by exploiting the LaSOT-based transfer set are shown in Table 3.9. The amount of training samples is lower than the amount obtained by filtering the GOT-10k transfer set with $\beta = 0.8$, and the proposed trackers present a behavior that follows the discussion presented in the previous paragraph. This experiment confirms that the quantity of data has more impact than its quality on the student.

3.4.3 State-of-the-Art Comparison

We now compare the best configurations of the proposed tracking modalities against the state-of-the-art.

Table 3.10 shows that TRASFUST, on the GOT-10k benchmark, competes with the most recent trackers like Ocean. TRASFUST outperforms MEEM whose methodology builds up on the experience of other trackers. TRAST perform comparably to other recent methods like D3S and SiamCAR. TRAS tracks better than ROAM, SiamFC, and the deep regression tracker GOTURN.

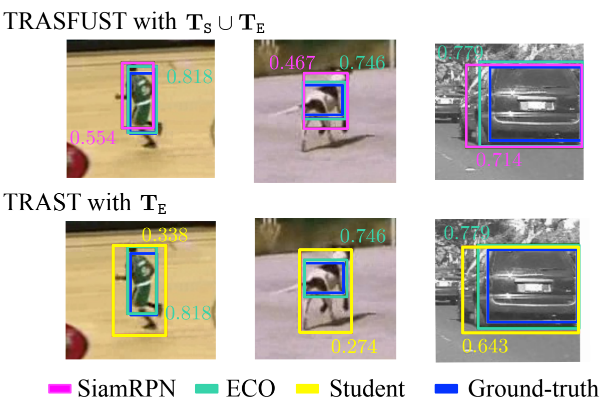


Figure 3.8: Behavior of TRASFUST and TRAST with ambiguous ground-truths. In the presented frames, TRASFUST selects the bounding-box predicted by $\mathbf{T}_S$, while TRAST the one of the student. These outputs are qualitative better but have much less IoU (quantified by the colored numbers) with respect to $b_t^{(g)}$. This impacts the overall quantitative performance. (Figure better analyzed in color).

Table 3.6: Performance of the proposed tracking strategies with and without the learning components introduced by our methodology. Best benchmark results, per tracker, are highlighted in bold.

	GOT-10k			UAV123		LaSOT		OTB-100	
	AO	$SR_{0.50}$	$SR_{0.75}$	SS	PS	SS	PS	SS	PS
TRASFUST	**0.519**	**0.616**	**0.287**	**0.628**	**0.823**	**0.510**	**0.505**	**0.675**	**0.899**
TRASFUST-no-curr	0.506	0.599	0.278	0.627	0.819	0.496	0.484	0.665	0.879
TRAST	**0.531**	0.626	0.345	**0.603**	**0.773**	**0.490**	**0.470**	**0.604**	**0.818**
TRAST-no-curr	0.530	**0.630**	**0.347**	0.602	0.770	0.484	0.464	0.595	0.794
TRAS	**0.484**	**0.556**	**0.326**	**0.515**	**0.655**	**0.386**	**0.330**	**0.481**	**0.644**
TRAS-no-curr	0.474	0.547	0.307	0.501	0.644	0.385	0.323	0.447	0.600
TRAS-KD	0.422	0.481	0.239	0.494	0.634	0.340	0.276	0.457	0.635
TRAS-KD-GT	0.448	0.499	0.305	0.491	0.630	0.354	0.298	0.448	0.606
TRAS-GT	0.444	0.495	0.286	0.483	0.616	0.331	0.271	0.438	0.581

The results achieved on the UAV123 benchmark, presented in Table 3.11, show that TRASFUST competes with PrDiMP which currently holds the best SS performance. TRAST performs on par with Siam-R-CNN and SiamBAN, while TRAS outperforms methods like GCT and GOTURN.

On the LaSOT benchmark (Table 3.12) TRASFUST performs better than Ocean but lower than PrDiMP. TRAST results stronger than PG-Net, SiamBAN, and SiamCAR. TRAS has better performance than GradNet and than the deep regression tracker RE3 which uses a similar CNN architecture but a different learning strategy.

TRASFUST achieves the SS performance of Siam-R-CNN on the OTB-100 dataset, but with an increased PS that surpasess also the one of Ocean, SiamBAN and C-COT. TRAST performs on par with GradNet, while TRAS outperforms RE3.

A comparison was also done using the recent VOT2020 benchmark. For this, all the trackers have been set to output the segmentation mask for the target objects through the SiamSeg method [131]. The results achieved show that TRASFUST outperforms the

Table 3.7: Performance of the three tracking modalities while considering different teacher setups for training. Best teacher configuration, per tracker and benchmark, is highlighted in bold.

	Teachers	GOT-10k			UAV123		LaSOT		OTB-100	
		AO	$SR_{0.50}$	$SR_{0.75}$	SS	PS	SS	PS	SS	PS
TRASFUST	$\mathbf{T}_{\mathtt{K}}$	0.513	0.607	0.295	0.632	0.823	0.514	0.505	0.678	0.896
	$\mathbf{T}_{\mathtt{M}}$	0.517	0.611	0.294	0.632	0.827	0.512	0.503	0.681	**0.908**
	$\mathbf{T}_{\mathtt{E}}$	0.511	0.605	0.293	**0.638**	**0.832**	0.507	0.499	0.681	0.903
	$\mathbf{T}_{\mathtt{S}}$	**0.522**	**0.619**	**0.305**	0.637	0.825	**0.516**	**0.507**	**0.682**	0.905
	$\mathbf{T}_{\mathtt{P}}$	0.519	0.616	0.287	0.628	0.823	0.510	0.505	0.675	0.899
TRAST	$\mathbf{T}_{\mathtt{K}}$	0.390	0.440	0.191	0.526	0.682	0.388	0.319	0.495	0.660
	$\mathbf{T}_{\mathtt{M}}$	0.452	0.521	0.223	0.572	0.776	0.433	0.386	0.569	0.793
	$\mathbf{T}_{\mathtt{E}}$	0.491	0.571	0.249	0.580	0.768	0.442	0.397	0.583	0.786
	$\mathbf{T}_{\mathtt{S}}$	**0.532**	**0.632**	**0.354**	**0.605**	**0.779**	0.485	0.457	0.601	0.806
	$\mathbf{T}_{\mathtt{P}}$	0.531	0.626	0.345	0.603	0.773	**0.490**	**0.470**	**0.604**	**0.818**
TRAS	$\mathbf{T}_{\mathtt{K}}$	0.371	0.418	0.178	0.464	0.598	0.321	0.241	0.390	0.524
	$\mathbf{T}_{\mathtt{M}}$	0.414	0.473	0.214	0.462	0.606	0.336	0.262	0.390	0.545
	$\mathbf{T}_{\mathtt{E}}$	0.422	0.484	0.232	0.507	0.652	0.357	0.286	0.422	0.567
	$\mathbf{T}_{\mathtt{S}}$	0.441	0.499	0.290	**0.517**	0.646	0.377	0.310	0.447	0.599
	$\mathbf{T}_{\mathtt{P}}$	**0.484**	**0.556**	**0.326**	0.515	**0.655**	**0.386**	**0.330**	**0.481**	**0.644**

Table 3.8: Performance of the proposed trackers considering $\mathbf{T}_{\mathtt{P}}$'s increasingly better predictions on the GOT-10k-based transfer set. Best threshold results, per tracker and benchmark, are highlighted in bold.

	Tracker	GOT-10k			UAV123		LaSOT		OTB-100	
		AO	$SR_{0.50}$	$SR_{0.75}$	SS	PS	SS	PS	SS	PS
$\beta = 0.5$	TRASFUST	**0.519**	**0.616**	0.287	0.628	0.823	0.510	0.505	0.675	0.899
	TRAST	**0.532**	**0.632**	**0.354**	**0.605**	**0.779**	**0.485**	**0.457**	0.601	0.806
	TRAS	**0.484**	**0.556**	**0.326**	**0.515**	**0.655**	**0.386**	**0.330**	**0.481**	**0.644**
$\beta = 0.6$	TRASFUST	0.507	0.599	**0.295**	**0.639**	**0.827**	**0.514**	**0.510**	**0.683**	**0.901**
	TRAST	0.518	0.616	0.326	0.599	0.768	0.475	0.452	**0.608**	**0.809**
	TRAS	0.426	0.488	0.244	0.481	0.609	0.343	0.277	0.452	0.617
$\beta = 0.7$	TRASFUST	0.507	0.599	0.289	0.638	**0.827**	0.513	0.505	0.675	0.894
	TRAST	0.513	0.603	0.310	0.594	0.766	0.478	0.456	0.586	0.781
	TRAS	0.404	0.449	0.231	0.430	0.552	0.334	0.260	0.390	0.522
$\beta = 0.8$	TRASFUST	0.494	0.575	0.260	0.624	0.815	0.494	0.482	0.672	0.888
	TRAST	0.505	0.598	0.297	0.592	0.764	0.457	0.426	0.589	0.774
	TRAS	0.326	0.344	0.155	0.387	0.489	0.243	0.170	0.323	0.414
$\beta = 0.9$	TRASFUST	0.403	0.425	0.169	0.534	0.743	0.401	0.374	0.626	0.836
	TRAST	0.471	0.541	0.250	0.547	0.697	0.445	0.409	0.574	0.746
	TRAS	0.140	0.070	0.014	0.064	0.045	0.086	0.019	0.132	0.104

trackers DiMP, ECO, and ATOM, while TRAST performs comparably to SiamRPN. TRAS exhibits some difficulties on this benchmark but its A performance is still higher than that of KCF. It is interesting to notice how the use of more teachers during tracking increases the R score of the tracking modalities. Considering the results of the annual VOT challenge [119], we had that the instance of the methodology presented in this chapter – TRASFUSTm – resulted the 10th best tracker in the short-term challenge, while the TRASTmask instance resulted 10th in the real-time challenge.

3.5 Conclusions

In this chapter, a new deep learning methodology for visual object tracking has been proposed. The underlying idea was to use trackers as efficient building blocks for both

Table 3.9: Performance of the proposed trackers considering the training set of LaSOT as transfer set.

	Tracker	GOT-10k			UAV123		LaSOT		OTB-100	
		AO	$SR_{0.50}$	$SR_{0.75}$	SS	PS	SS	PS	SS	PS
$\beta = 0.5$	TRASFUST	0.468	0.529	0.221	0.594	0.803	0.470	0.452	0.666	0.885
	TRAST	0.475	0.552	0.248	0.553	0.746	0.463	0.432	0.577	0.760
	TRAS	0.242	0.252	0.086	0.329	0.437	0.222	0.166	0.254	0.337

Table 3.10: Performance of the proposed trackers against the state-of-the-art on the GOT-10k test set [39]. Best tracker, per measure, is highlighted in red, second-best in blue.

	MEEM [99]	GOTURN [41]	SiamFC [43]	ROAM [122]	SiamCAR [123]	D3S [124]	Ocean [98]	TRAS	TRAST	TRASFUST
AO	0.253	0.347	0.348	0.436	0.569	0.597	**0.611**	0.484	0.604	**0.617**
$SR_{0.50}$	0.235	0.375	0.353	0.466	0.670	0.676	**0.721**	0.556	0.708	**0.729**
$SR_{0.75}$	0.068	0.124	0.098	0.164	0.415	0.462	-	0.326	**0.469**	**0.490**

tracking and learning. To this end, the KD and RL paradigms were joined in a novel framework where off-the-shelf state-of-the-art tracking algorithms are exploited. Tracking knowledge is first transferred offline from teacher trackers to a CNN-based student model. After a single end-to-end learning stage, student and teacher are used by a tracker that can be derived in three different modalities, namely TRASFUST, TRAST and TRAS, that achieve the goals of tracker fusion, target adaptation, and fast processing speed. An extensive experimental validation showed that TRASFUST competes with the most recent state-of-the-art solutions, while TRAST and TRAS compete with the respective class of state-of-the-art methods. All the modalities can run in real-time.

Table 3.11: Performance of the proposed trackers against the state-of-the-art on the UAV123 benchmark [70]. Best tracker, per measure, is highlighted in red, second-best in blue.

	GOTURN [41]	GCT [125]	SiamCAR [123]	SiamBAN [97]	Siam-R-CNN [126]	PrDiMP [127]	TRAS	TRAST	TRASFUST
SS	0.389	0.508	0.614	0.631	0.649	**0.680**	0.515	0.647	**0.679**
PS	0.548	0.732	0.760	0.833	0.834	-	0.655	**0.837**	**0.873**

Table 3.12: Performance of the proposed trackers against the state-of-the-art on the LaSOT test set [38]. Best tracker, per measure, is highlighted in red, second-best in blue.

	RE3 [42]	GradNet [128]	SiamCAR [123]	SiamBAN [97]	PG-Net [129]	Ocean [98]	PrDiMP [127]	TRAS	TRAST	TRASFUST
SS	0.325	0.365	0.507	0.514	0.531	0.560	**0.598**	0.386	0.545	**0.576**
PS	0.301	0.351	0.510	-	-	**0.566**	-	0.330	0.524	**0.574**

Table 3.13: Performance of the proposed trackers against the state-of-the-art on the OTB-100 benchmark [71]. Best tracker, per measure, is highlighted in red, second-best in blue.

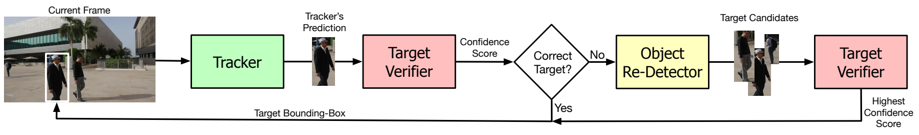
	RE3 [42]	GradNet [128]	C-COT [59]	Ocean [98]	KYS [130]	SiamBAN [97]	Siam-R-CNN [126]	TRAS	TRAST	TRASFUST
SS	0.464	0.639	0.673	0.684	0.695	**0.696**	**0.701**	0.481	0.643	**0.701**
PS	0.582	0.861	0.903	**0.920**	-	0.910	0.891	0.644	0.865	**0.931**

Table 3.14: Performance of the proposed trackers against the state-of-the-art on the VOT2020 benchmark [119]. All results are achieved by considering the SiamSeg method [131] as target segmentation generator. Best tracker, per measure, is highlighted in red, second-best in blue.

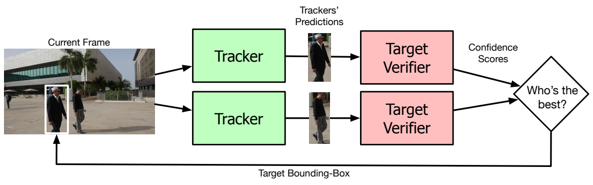
	KCF [32]	SiamFC [43]	MetaCrest [132]	SiamRPN [44]	ECO [60]	ATOM [61]	DiMP [62]	TRAS	TRAST	TRASFUST
EAO	0.285	0.309	0.336	0.369	**0.414**	0.406	0.410	0.238	0.370	**0.424**
A	0.569	0.682	0.657	**0.701**	0.694	0.691	0.691	0.678	0.684	**0.696**
R	0.501	0.571	0.624	0.651	0.729	0.723	**0.730**	0.450	0.677	**0.745**

4 Combining Trackers in the Long-Term Context

Chapter 3 presented a visual tracking methodology based on the exploitation of other trackers. One of the particular tracking modalities proposed in such a framework was TRASFUST, a tracker which implemented a tracker decision strategy to fuse the capabilities of complementary trackers in order to achieve increased accuracy. Despite the good results achieved, such a solution was designed to work in short-term tracking scenarios. As we discussed in the introduction of this Thesis, depending on the behavior of the target and the dynamics of the captured scene, a visual tracking problem can be either divided into short-term tracking or long-term tracking [133]. The first occurs when the target never leaves completely the camera's field of view. This is the most popular setting represented by the community's benchmark datasets [71, 92, 70, 134, 135] and subsequently the most tackled by the solutions available at the state-of-the-art. Indeed, successful methodologies include deep discriminative trackers [61, 62], deep siamese networks [43, 45, 136], deep regression trackers [41, 42], and more recently transformers [66, 65, 67]. In the setting of long-term tracking problems the assumption of the target being always visible is relaxed. In such scenarios the object is permitted to disappear by leaving the field of view or by being completely hidden by another object. These situations require a tracker to produce not only the target's localization but also a confidence score expressing wether the object is visible or not [133]. In the past, long-term trackers [138, 139, 140, 137] consisted in variations of an essential scheme (see Figure 4.1(a)) composed of: a short-term tracking algorithm to follow the target while visible; a re-detection operation to find again the target after its disappearance; a verification module to check if the short-term tracker and the re-detector have localized the object of interest. More recently, the properties of such complex and often inefficient pipelines are starting to be achieved with compact deep neural network-based trackers. Indeed, new visual trackers such as the deep discriminative tracker SuperDiMP [62, 119] and the transformer-based solution STARK [67] are able to match or even surpass the performance of previous methodologies while performing at real-time speed. Such an efficiency



(a)



(b)

Figure 4.1: Comparison of the schemes of (a) the most successful systems for long-term visual tracking exploited in the last years [137, 92, 119], (b) our proposed system. In this study, we tackle the long-term tracking setting by proposing a lightweight framework in which the tracking performance of two algorithms run in parallel is evaluated and selected for improved target localization.

makes the employment of tracker fusion strategies [141, 100, 101, 142] appealing since the processing speed of an ensemble-based solution could be reasonably good for applications. Such class of approaches, including our TRASFUST, demonstrated how increased tracking performance can be achieved by the careful combination of the complementary capabilities of different trackers, as are those of SuperDiMP and STARK (see Figure 4.3). However, the available solutions focused on the fusion of trackers in short-term scenarios and, to the best of our knowledge, yet no work explored such strategies in the context of long-term visual object tracking.

Hence, in this chapter, we try to fill such a gap by proposing a methodology that combines the capabilities of baseline trackers for long-term tracking problems. Our strategy is based on a procedure of tracker evaluation which determines if each of the baseline trackers is correctly tracking the target. This is achieved by an online learned deep neural network able to distinguish the target object from the surrounding background. The proposed evaluation strategy allows to select the best target localization proposed by the trackers. The outcome of the selection is exploited to make the trackers interact and correct their performance during tracking. Our proposed solution improves the performance of the underlying trackers by a good margin. New state-of-the-art results are achieved on the LTB-35 [73], LTB-50 [133, 92], TLP [143], and LaSOT benchmarks [38]. Our methodology – referred to as mlpLT – was awarded as the "winning tracker" of the Visual Object Tracking VOT2021 Long-Term Challenge [144], the most relevant challenge for long-term visual tracking solutions. In addition to these outcomes, in this chapter we provide additional experimental results that explain the behavior of our solution.

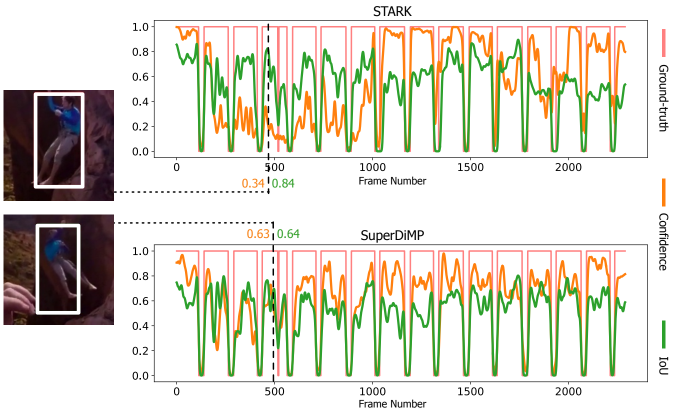


Figure 4.2: This figure shows the complementary characteristics of the STARK [67] and SuperDiMP [62] trackers along a video sequence. The first solution has a better ability in producing bounding-boxes that tightly fit the target's appearances, leading to an higher average overlap but that is not consistent with its confidence trend. The second solution is less accurate in terms of bounding-box overlap but its confidence predictions are more consistent with its performance, ultimately demonstrating more robustness.

4.1 Related Work

In this section we review the most relevant works related to our proposed methodology.

4.1.1 Long-Term Visual Tracking

Kalal et al. [138] considered the long-term tracking task under the fruitful framework of tracking, learning, and detection in which: a short-term tracker based on median-flows follows the target while visible; a learning module generates training examples during tracking for target recognition; and an online learned cascade classifier is used as target detector. Such a scheme has been then improved by many follow-up solutions [140, 137] which exploited deep learning models to implement the short-term tracker, the target verification module, or the detector. Differently from such approaches, the FuCoLoT tracker [145] extended the discriminative correlation filter (DCFs) approach [31] to the long-term setting by optimizing multiple filters at different time scales to implement a short-term tracker and a long-term detector which predictions are then fused together. The GlobalTrack tracker [146] made the deep siamese approach [43] work in long-term scenarios by searching the target globally instead of locally in each frame. More recently, the STARK tracker [67] exploited transformer neural networks [64] to implement an effective matching operation that is able to perform short-term tracking and re-detection at the same time.

Enlightened by the recent capabilities of trackers which achieve remarkable results in the long-term context without sacrifying efficiency [67, 62, 119], in this work we follow a different idea with respect to those introduce in this section and propose a new framework for long-term tracking that aims to combine the characteristics of complementary tracker with an effective decision and interaction strategy.

4.1.2 Tracker Fusion Strategies

Different approaches have been proposed to fuse the execution of multiple trackers while tracking. Yoon et al. [141] made different trackers interact by exploiting a probabilistic approach based on particle filters. The MEEM tracker [99] later provided a multi-expert framework where trackers are fused via a procedure based on entropy minimization. Wang et al. [147] and Vojir et al. [101] used variations of Hidden Markov models to implicitly correct an ensemble of interactive trackers. Bailer et al. [100] used an optimization approach based on dynamic programming to fuse the predictions of multiple trackers without their interaction. For an analogous setting, Dunnhofer et al. [142] presented a tracker selection strategy based on a value function approximation learned offline via knowledge distillation and reinforcement learning (RL). Similarly, Song et al. [148] proposed an online selection policy optimized with hierarchical RL.

The main drawback of the solutions presented here is that they were studied for the fusion of trackers in the context of short-term tracking. In contrast, in this chapter we study a solution for the long-term setting and, to the best of our knowledge, this is new in such a context.

4.2 Methodology

The key idea of this Chapter is to develop an effective strategy to fuse the capabilities of complementary trackers in the context of long-term visual object tracking. Particularly, our goal is to implement a solution that achieves higher tracking performance in an online fashion by exploiting the characteristics of different trackers. After the description of some preliminary concepts, in this section we will introduce the methodology to accomplish such an objective.

4.2.1 Preliminaries

We consider a video $\mathcal{V} = \{F_t \in \mathcal{I}\}_{t=0}^{T}$ as a sequence of frames F_t, where $\mathcal{I} = \{0, \cdots, 255\}^{w \times h \times 3}$ is the space of RGB images and $T \in \mathbb{N}$ denotes the number of frames. Let $b_t = [x_t, y_t, w_t, h_t] \in \mathcal{B} \subseteq \mathbb{R}^4$ be the t-th bounding-box defining the coordinates of the top left corner, and the width and height of the rectangle containing the target. The goal of a long-term tracker is to predict the bounding-box b_t that best fits the target and a confidence score $c_t \in [0, 1]$ that reports whether the target is visible in the frame, for all F_t. We define a tracker as a function $\tau : \mathcal{I} \to \mathcal{B} \times [0, 1]$ that returns the target localization and confidence score for an input frame. At the first frame F_0, the tracker is initialized with the ground-truth bounding-box b_0 which outlines the target to be tracked.

4.2.2 Tracker Combination Procedure

We refer to our proposed combination algorithm as $\tau^{(ours)}$. At every t, $\tau^{(ours)}$ receives in input the frame F_t and outputs b_t and c_t. The details about the procedure implemented by $\tau^{(ours)}$ are given as pseudo-code in Algorithm 3.

Execution of The Baseline Trackers

In this study, we selected the state-of-the-art trackers STARK [67] and SuperDiMP [62, 119] enhanced with a meta-updater [137] as the baseline trackers which capabilities are fused. Such choices are motivated by the outstanding results achieved by such algorithms in the long-term setting, and because they perform tracking by different principles. Indeed, the first method is based on a transformer-based architecture [64] whose tracking knowledge is acquired on a large dataset of tracking examples only through offline optimization. The second tracker instead is a deep discriminative tracker which uses an online learning mechanism to adapt a pretrained network to a new target while tracking. The aspect of performing tracking by disparate principles is especially important since complementary capabilities could benefit a combination strategy. We verified the presence of such complementary characteristics in the long-term tracking behavior of the considered trackers, and an example of outcome is proposed in Figure 4.3. It can be noticed that STARK manifests a better ability in producing bounding-boxes tightly fitting the targets' appearance. This behavior results in an higher Intersection-over-Union (IoU) and demonstrates that the tracker is more spatially accurate. Such an ability however is not consistent with the confidence predictions given by the tracker which are often wrong or overconfident. On the other hand, we observe that SuperDiMP is generally less accurate in the prediction of target localization – its IoU is lower than STARK – but its confidence predictions are definitely more consistent with such a performance, ultimately demonstrating an increased robustness. For a better explanation, from now on we refer to STARK as $\tau^{(1)}$ and to SuperDiMP as $\tau^{(2)}$.

In our proposed pipeline $\tau^{(ours)}$, such two baseline trackers $\tau^{(1)}, \tau^{(2)}$ are run according to their original methodology on frame F_t when $\tau^{(ours)}$ is inputted with frame F_t. By this step, they produce the respective bounding-box $b_t^{(i)}$ and confidence score $c_t^{(i)}$, $i = 1, 2$. It is worth notice that the two trackers are one independent from the other. It is hence possible to put in parallel the executions of the two in order to increase the processing speed of the combination strategy.

Target Visibility Determination

Next, $\tau^{(ours)}$ determines whether $\tau^{(i)}$ are correctly following the target, i.e. if it is visible in their predicted bounding-boxes. This step is achieved by exploiting the confidence $c_t^{(i)}$ which represent tracker-specific probability estimates of the target being present in the frame. However, relying solely on $c_t^{(i)}$ does not enable an effective tracker selection mechanism because such estimates can be erroneous due to overconfidence or training bias. We hence propose to improve such target visibility scores through a target verification module that is independent from the baseline trackers. Particularly, we employ an online learned function $v : \mathcal{I} \times \mathcal{B} \rightarrow [0, 1]$ that returns a probability estimate v_t of the target being present in the image patch extracted from the frame F_t considering the area determined by a bounding-box b_t. This operation is inspired by the target verification operation present in different long-term tracking pipelines [137, 92, 149]. Such a verification step is implemented as a binary classification based on a deep neural network learned to distinguish between patches containing the target object and patches

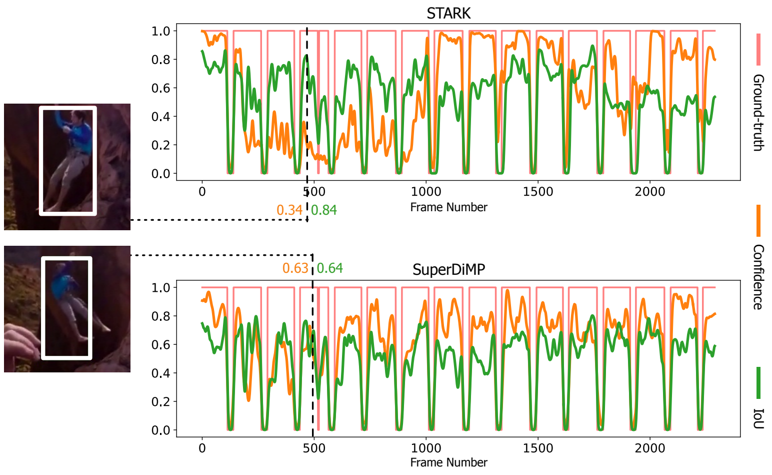


Figure 4.3: This figure shows the complementary characteristics of the STARK [67] and SuperDiMP [62] trackers along a sequence. The first has a better ability in producing bounding-boxes better aligned with the target but that is not consistent with its ability in target presence confidence. The second provides less precise target localizations but that are more consistent with the confidence predictions.

without. The architecture and learning procedure is akin to [40]. In short, the network is first trained offline to acquire general patch separation knowledge. During tracking, the pretrained weights are adjusted online by an optimization procedure that uses new target appearances extracted based on the latest target localization information b_t. A sampling procedure is performed to generate candidate target localization around b_t. Of such samples, positive target patches are those image areas whose sampled location has an IoU greater than 0.7. Negative target patches are those resulting in an IoU of 0.3 or lower instead. The execution of this update operation is triggered by a meta updater instance [137].

Hence, in the proposed pipeline $\tau^{(ours)}$, the verifier $v(\cdot)$ takes the frame F_t and bounding-box $b_t^{(i)}$, and returns a tracker-independent evaluation score $v_t^{(i)}$ for each tracker. Such a value is combined with the tracker's confidence as

$$\widehat{c}_t^{(i)} = \frac{c_t^{(i)} + v_t^{(i)}}{2}. \tag{4.1}$$

$\widehat{c}_t^{(i)}$ represents a more consistent target visibility estimation. We binarize such values with a 0.5 threshold to determine the status $\widehat{p}_t^{(i)} \in \{0, 1\}$ of target visual presence in the single frames. However, we experienced that $\widehat{p}_t^{(i)}$ could be wrong since the estimation is mostly based on the single-frame appearance of targets. Given that target disappearances and reappearances are dynamic processes evolving over multiple frames we consider the $\widehat{p}_t^{(i)}$ present in the last $\widehat{T}$ frames to determine the actual target presence.

Algorithm 3 Pseudo-code of the procedure implemented by the proposed tracker fusion method $\tau^{(ours)}$.

```
// Consider video $\mathcal{V}$ and ground-truth $b_0^{(g)}$
// Assume learned weights $\theta$
$b_0 \leftarrow b_0^{(g)}$
$t \leftarrow 1$
// Baseline Trackers Initialization
Initialize $\tau^{(1)}$ with $F_0$ and $b_0$
Initialize $\tau^{(2)}$ with $F_0$ and $b_0$
repeat
    // Baseline Trackers Execution
    $b_t^{(1)}, c_t^{(1)} \leftarrow \tau^{(1)}(F_t)$ // Run the Stark tracker
    $b_t^{(2)}, c_t^{(2)} \leftarrow \tau^{(2)}(F_t)$ // Run the SuperDiMP tracker

    // Target Visibility Determination
    for $i = 1, 2$ do
        $v_t^{(i)} \leftarrow v(F_t, b_t^{(i)})$
        $\widehat{c}_t^{(i)} \leftarrow \frac{c_t^{(i)} + v_t^{(i)}}{2}$
        $\widehat{p}_t^{(i)} \leftarrow \widehat{c}_t^{(i)} \geq 0.5$
        $p_t^{(i)} \leftarrow \sum_{j=0}^{\widehat{T}} \widehat{p}_{t-j}^{(i)} > \lfloor 0.75 \cdot \widehat{T} \rfloor$
    end for

    // Target Localization Determination
    $\widehat{i} \leftarrow \arg\max_i (p^{(i)})$
    if $p_t^{\widehat{i}} = 1$ then
        $b_t \leftarrow b_t^{(\widehat{i})}$

        // Tracker Correction
        Correct the other tracker position with $b_t$
    else
        $b_t \leftarrow b_t^{(1)}$
    end if
    $c_t \leftarrow p_t$
    Return $b_t, c_t$ as output
    $t \leftarrow t + 1$
until $t = T$
```

Particularly, we say that the target is visible inside $b_t^{(i)}$, and set $p_t^{(i)} = 1$, if

$$\sum_{j=0}^{\widehat{T}} \widehat{p}_{t-j}^{(i)} > \lfloor 0.75 \cdot \widehat{T} \rfloor. \tag{4.2}$$

Otherwise, we set $p_t^{(i)} = 0$ and the target is considered not visible in the tracker's prediction. We experimentally found $\widehat{T} = 5$ to be a good representation of the duration of the target disappearance/appearance process. The value $p_t^{(i)}$ is also used as confidence prediction of the proposed $\tau^{(ours)}$, i.e. $c_t = p_t^{(i)}$.

Target Localization Determination

The values $p_t^{(i)}$ determine which tracker is currently following the target. Such information is used by $\tau^{(ours)}$ to select which target bounding-box to output for frame F_t. If $p_t^{(1)}$ and $p_t^{(2)}$ are equal and greater than 0 we select the box produced by $\tau^{(1)}$ as output b_t since it produces more accurate bounding-boxes in general. If only one of the two $p_t^{(i)}$ values is equal to 1 then the tracker's box corresponding to i is determined and used as output. If both $p_t^{(i)}$ are zero, the bounding-box result of $\tau^{(1)}$ is selected because of its better re-detection capabilities.

Tracker Correction

The predicted b_t is an useful resource if properly aligned on the target. We hence exploit it to correct the performance of the worse tracker. At every F_t, $\tau^{(1)}$ and $\tau^{(2)}$ search for the target in an image area determined by the previously known bounding-box $b_{t-1}^{(i)}$. We propose to modify such target position to match b_{t-1} when $p_t^{(\hat{i})} = 1$. This step has a correction effect on the behavior of $\tau^{(i)}$. In fact, at the next frame F_{t+1}, $\tau^{(i)}$ will search for the target in a local image area whose position is more consistent with the target position in the current frame.

4.3 Experimental Setup

In this section, we describe the settings employed to evaluate the proposed methodology.

4.3.1 Datasets

We conducted experiments on the LTB-35 [73] and LTB-50 [133] benchmarks. These are datasets used in the annual VOT challenges [73, 92, 119]. They are composed of 35 and 50 videos, respectively, densely labeled with the bounding-boxes of diverse objects (people, car, motorcycles, bicycles, boat, animals, etc.). LTB-50, which extends LTB-35 by 15 sequences, is composed of around 215K frames and each video contains circa 10 long-term target disappearances on average each lasting for circa 52 frames.

Evaluations were also performed on the "Track Long and Prosper" (TLP) dataset [143]. This benchmark is composed of 50 video sequences comprising around 676K labeled frames. The average length of the sequences in time is over 8 minutes.

We also ran experiments on the test set of the LaSOT benchmark [38]. This is composed of 280 sequences with around 690K frames and an average sequence length of 2500 frames. Target objects appearing in this dataset belong to 70 different categories.

4.3.2 Evaluation Protocol and Measures

For all the experiments over all the benchmarks, we run trackers according to the standard protocol [133, 71, 92] of initializing the tracker in the first frame and then execute it on every other frame to obtain bounding-box and confidence predictions. We employed established metrics to quantify the performance of our proposed solution. For the LTB-35 and LTB-50 benchmarks, the F-score, Precision_{LTB}, and Recall_{LTB} metrics [133]

Table 4.1: Performance achieved by the proposed solution on the LTB-50 benchmark [133, 92] with different coonfiguration settings. The first two rows report the performance of the underlying trackers STARK [67] and SuperDiMP [62, 119]. Best setup, per metric, is highlighted in bold.

Setup	F-Score	Precision_{LTB}	Recall_{LTB}
1) SuperDiMP	0.671	0.691	0.652
2) STARK	0.696	0.708	0.685
3) $\tau^{(ours)}$ exploiting only maximum $v_t^{(i)}$	0.706	0.697	0.715
4) $\tau^{(ours)}$ exploiting $\widehat{c}_t^{(i)}$ and no correction	0.710	0.709	0.711
5) $\tau^{(ours)}$ exploiting $\widehat{c}_t^{(i)}$	0.719	0.724	0.714
6) $\tau^{(ours)}$ exploiting $p_t^{(i)}$ and no correction	0.712	0.714	0.712
7) $\tau^{(ours)}$ exploiting $p_t^{(i)}$	0.722	0.728	0.717
8) + adaptive searching area	0.730	0.741	0.719
9) + aspect-ratio correction	**0.735**	**0.741**	**0.729**

have been used. On the TLP dataset, we employed the Area-Under-the-Curve (AUC) of the success plot – referred to as Success_{TLP} – in which an IoU of 1 is set for all the frames where the tracker correctly predicts the absence of the target. Similarly, we also report the Precision_{TLP} which is the AUC of the precision plot in which a bounding-box distance of 0 is set for all the frames where the tracker correctly predicts the absence of the target [143]. For the LaSOT benchmark, we used the AUC of the success and the precision plots referred as Success_{LaSOT} and Precision_{LaSOT} respectively [38].

4.3.3 Improvements to STARK

We found additional improvements to the tracking strategy of the underlying trackers to benefit the performance of our overall solution $\tau^{(ours)}$. In particular, we propose to control STARK's searching area factor σ which defines the image area size in which to look for the target. We considered $\sigma = \frac{\sigma}{2}$ in all frames in which $p_t^{(\widehat{i})} = 1$. Given that p_t establishes that $\tau^{(ours)}$ is following the visible target, the proposed improvement forces the tracker to better focus on it, reducing the chance of confusion due to the presence of distractors. Moreover, we found STARK to be susceptible to wrong target size estimations after the change of the dynamic template. We propose to penalize the results of STARK by setting $c_t^{(1)} = 0$ if the ratio between the aspect-ratio of b_{t-1} and $b_t^{(1)}$ are not consistent with the temporal coherence of motion and scale change of a target. Overall, as we will show later, these improvements permit to avoid wrong target image patches to pollute the training data used by $v(\cdot)$ for online adaptation, ultimately making its discriminative ability more effective.

4.3.4 Implementation Details

Code to implement the method and the experiments was implemented in Python and run on a machine with an Intel Xeon E5-2690 v4 @ 2.60GHz CPU, 320 GB of RAM and an NVIDIA TITAN V GPU. The original implementations of the STARK and SuperDiMP trackers provided by the respective authors have been used along with the pretrained models. The verifier model $v(\cdot)$ has been implemented using the PyTorch version of the MDNet tracker [40].

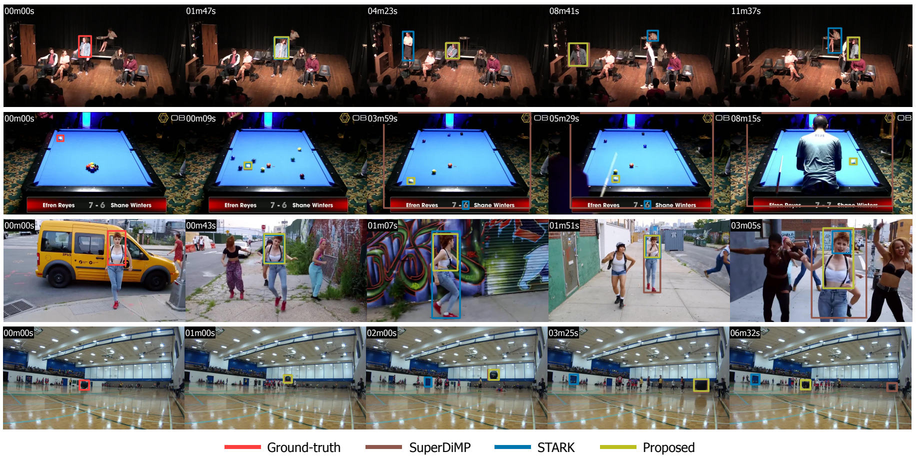


Figure 4.4: Qualitative examples of the tracking ability achieved by the proposed algorithm in comparison with the baseline trackers STARK and SuperDiMP. The first column of images presents the first frame of each video. In the top-left corner of each frame the time elapsed since the beginning of the video is reported. Overall, our solution permits to fuse the capabilities of the underlying trackers and consequently achieve a more robust target tracking along the videos.

4.4 Results

In this section, we provide the results of the conducted experimental campaign. We first analyze the capabilities of the proposed strategy on the LTB-50 benchmark [133, 92] under different ablative studies and in comparison with other tracker fusion strategies. We then compare our framework with many state-of-the-art solutions on the other benchmarks described in Section 4.3.

4.4.1 Ablation Study

Table 4.1 reports the performance of $\tau^{(ours)}$ on the LTB-50 benchmark [133, 92] by increasingly adding the contributions described in the Section 4.2. Improved performance with respect to the underlying trackers is already achieved by selecting the tracker obtaining the best $v_t^{(i)}$ given by the verifier (row 3). Precision$_{LTB}$ results are particularly increased by combining the trackers' confidences and the verifier scores (row 4). Determining the target presence $p_t^{(i)}$ by the scores achieved in the previous $\widehat{T}$ frames additionally increases the precision (row 6). Row 5 and 7 show the performance is particularly improved by the interaction and correction process between trackers. By this setup, the performance gain with respect to the best of the underlying trackers is of around 3.7% in F-Score, 2.8% in Precision$_{LTB}$, and 4.7% in Recall$_{LTB}$. The introduction of the improvements to STARK enables the $v(\cdot)$ to remove polluted samples during the training set used for online learning (row 7 and 8). This results in a more consistent optimization process that ultimately enables a better target-background discrimination. Notice that the same strategies activated on the underlying STARK tracker based on its confidence

Table 4.2: Performance of the proposed tracker on the LTB-50 benchmark [133, 92] in comparison with baseline strategies that combine the capabilities of the STARK and SuperDiMP trackers (whose performances are reported in the first two rows). Best result, per metric, is highlighted in bold.

Setup	F-Score	Precision$_{LTB}$	Recall$_{LTB}$
1) SuperDiMP	0.671	0.691	0.652
2) STARK	0.696	0.708	0.685
3) $b_t^{(i)}$ and $c_t^{(i)}$ average	0.671	0.723	0.626
4) $b_t^{(i)}$ and $c_t^{(i)}$ average and correction of both	0.704	0.711	0.698
5) $b_t^{(i)}$ selection by maximum $c_t^{(i)}$	0.705	0.703	0.706
6) $b_t^{(i)}$ selection by maximum $c_t^{(i)}$ and correction of the other	0.675	0.687	0.675
7) TRASFUST [142]	0.693	0.701	0.684
8) $\tau^{(ours)}$	**0.735**	**0.741**	**0.729**

$c_t^{(1)}$ make it achieve an F-Score of 0.694 and 0.689 for the adapting searching area and aspect-ratio correction respectively. These outcomes suggest that such strategies have to be applied carefully only when the estimation of target presence is sufficiently accurate. Overall, the performance gain of our overall system with respect to the best of the underlying trackers is of 5.6% in F-Score, 4.7% in Precision$_{LTB}$, and 6.4% in Recall$_{LTB}$. Some qualitative examples of the performance of $\tau^{(ours)}$ in comparison with the baseline trackers are presented in Figure 4.4.

4.4.2 Comparison with Baselines

We compared the fusion strategy implemented by $\tau^{(ours)}$ with baseline and state-of-the-art tracker fusion approaches [142]. All the compared methods have been applied on top of the same STARK and SuperDiMP instances used in $\tau^{(ours)}$. The results are given in Table 4.2. $\tau^{(ours)}$ results much better than all the other strategies. Simply averaging the bounding-box coordinate values and respective confidence scores lowers the performance of the trackers (row 1). Correcting both trackers by their average target position and scale improves the performance of the two (row 2). Selecting the $b_t^{(i)}$ for target localization based on the maximum $c_t^{(i)}$ of each tracker allows to improve the performance again (row 3). But making the trackers interact in the latter setup results in a performance drop (row 4). We hypothesize this is due to the STARK's overconfidence given to bounding-box predictions having low accuracy which causes the correction of the other trackers with inaccurate boxes. The fusion performance of TRASFUST [142] does not allow for performance improvement of the two underlying trackers. This happens because such a fusion strategy is designed for short-term tracking settings.

4.4.3 Target Presence Determination

Table 4.3 reports the performance of $\tau^{(ours)}$ in determining the target presence in the frames of the LTB-50 benchmark [133, 92] in comparison with the underlying trackers. Specifically, for each frame in the dataset we compared the target presence label (0 or 1) with each tracker's presence label computed after thresholding at 0.5 the tracker's

Table 4.3: Performance achieved by the proposed solution in the determination of the presence of the target on the LTB-50 benchmark [133, 92] in comparison with SuperDiMP and STARK. Best result, per metric, is highlighted in bold.

Setup	Accuracy	Sensitivity	Specificity
SuperDiMP	0.828	0.811	**0.907**
STARK	0.861	0.860	0.811
$\tau^{(ours)}$	**0.925**	**0.937**	0.782

Table 4.4: Performance achieved by $\tau^{(ours)}$ on the LTB-50 benchmark [133, 92] while considering different number of frames $\widehat{T}$ to determine the visibility of the target. Best result, per metric, is highlighted in bold.

# Frames	1	2	5	10	20
F-Score	0.728	0.732	**0.735**	0.734	0.719
Precision$_{LTB}$	0.728	0.736	**0.741**	0.734	0.719
Recall$_{LTB}$	0.728	0.728	0.729	**0.731**	0.719

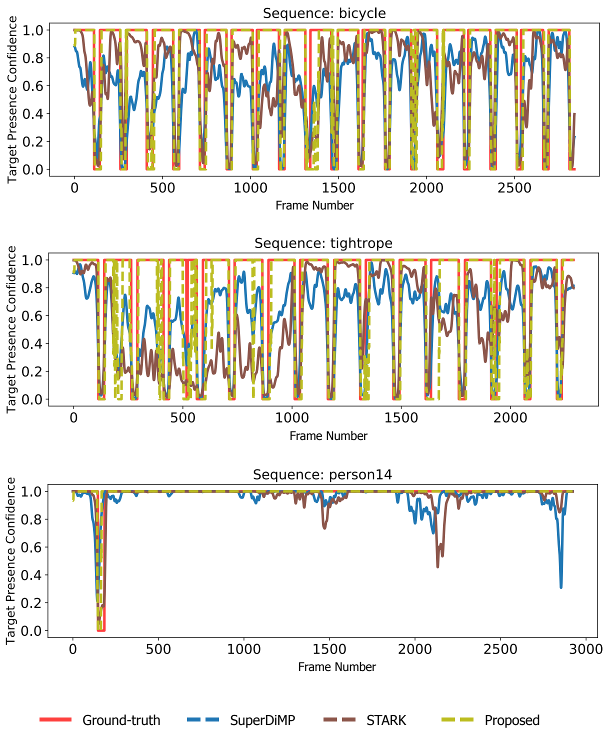


Figure 4.5: Examples of the ability of the proposed solution in determining the visual presence of the target (i.e the confidence) in the frames of three different sequences of the LTB-50 [133, 92]. The confidences predicted by STARK and SuperDiMP also reported for comparison. Our solutions results in a better and more stable estimation of target presence along the videos.

Table 4.5: Performance achieved by the proposed solution on the LTB-50 benchmark [133, 92] in comparison with the state-of-the-art. Best result, per metric, is highlighted in red, second-best in blue.

	SPLT	CLGS	DMTrack	LTMU_B	LT_DSE	STARK-ST50	KeepTrack	Zitong et al.	$\tau^{(ours)}$
F-Score	0.565	0.674	0.687	0.691	0.695	0.702	0.709	**0.711**	**0.735**
Precision_{LTB}	0.587	**0.739**	0.690	0.701	0.715	0.710	0.723	0.726	**0.741**
Recall_{LTB}	0.544	0.619	0.662	0.681	0.677	0.695	**0.697**	**0.697**	**0.729**

Table 4.6: Performance achieved by the proposed solution on the LTB-35 benchmark [73] in comparison with the state-of-the-art. Best result, per metric, is highlighted in red, second-best in blue.

	SPLT	Siam-R-CNN	SuperDiMP	Yu et al.	LTMU	STARK-ST50	KeepTrack	$\tau^{(ours)}$
F-Score	0.616	0.668	0.673	0.682	0.690	0.702	**0.720**	**0.739**
Precision_{LTB}	0.633	-	0.698	0.720	0.710	0.714	**0.738**	**0.749**
Recall_{LTB}	0.600	-	0.651	0.648	0.672	0.690	**0.704**	**0.729**

predicted $c_t^{(i)}$. The Accuracy reports the average agreement between the ground-truth and tracker-specific labels. The Sensitivity reports the fraction of correctly predicted presences in the frames where the target is actually present. The Specificity instead reports the fraction of correct non-presence predictions for those frames in which the target is not present. $\tau^{(ours)}$ presents a higher accuracy (+7.4%) and a higher sensitivity (+9%) with respect to the two trackers, meaning that it has a better ability in the determination of the target when this is visible. The Specificity is reduced, suggesting that, despite the overall good ability, the proposed strategy generates a larger amount of false positives during the absence of the target from the frames.

Figure 4.5 shows three examples of how the target presence estimation of our proposed methodology better fits with the ground-truth along a video sequence.

Table 4.4 reports the sensibility of the proposed target presence determination strategy in relation to the number of frames $\widehat{T}$. Employing a number of frames $\widehat{T} > 1$ is preferable but it must be assured that $\widehat{T}$ is not too large (i.e. > 10) since it could lead to a reduced tracking performance.

4.4.4 State-of-the-Art Comparison

In this paragraph, we present the performance of $\tau^{(ours)}$ in comparison with the state-of-the-art. For all the compared methodologies we report the performance presented in their original papers (the "-" symbol reports that the authors did not provide results for the particular measure).

Table 4.5 reports the results of $\tau^{(ours)}$ against the trackers SPLT [140], CLGS [92], DMTrack [150], LTMU_B [137], LT_DSE [137, 92], STARK-ST50 [67], KeepTrack [151], and the solution of Zitong2021. Our proposed tracker results the best solution across all the performance measures and hence sets new state-of-the-art results. The improvement over the second-best method [152] is of of 3.5% in F-Score, 2% in Precision_{LTB}, and of 4.6% in Recall_{LTB}. Our fusion strategy results better than the KeepTrack strategy [151], which augments the long-term capabilities of SuperDiMP by tracking and suppressing distractor objects, and even better than the LTMU_B pipeline [137] that uses the DiMP

Table 4.7: Performance achieved by the proposed solution on the TLP benchmark [143] in comparison with the state-of-the-art. Best result, per metric, is highlighted in red, second-best in blue.

	SiamFC	MDNet	Yu et al.	GlobalTrack	SuperDiMP	DMTrack	STARK-ST50	LTMU	$\tau^{(ours)}$
Success_{TLP}	0.237	0.365	0.488	0.520	0.537	0.541	0.549	**0.551**	**0.587**
Precision_{TLP}	0.281	0.384	-	0.567	0.563	0.591	0.568	**0.619**	**0.633**

Table 4.8: Performance achieved by the proposed solution on the LaSOT benchmark [38] in comparison with the state-of-the-art. Best result, per metric, is highlighted in red, second-best in blue.

	SuperDiMP	TrDiMP	LTMU	Siam-R-CNN	TransT	STARK-ST50	KeepTrack	$\tau^{(ours)}$
Success_{LaSOT}	0.631	0.639	0.647	0.648	0.649	0.664	**0.671**	**0.685**
Precision_{LaSOT}	0.653	0.663	0.665	0.684	0.690	0.693	**0.702**	**0.725**

tracker in the standard long-term scheme presented in Figure 4.1(a).

Similar outcomes are achieved for the subset LTB-35 [73]. In this case, $\tau^{(ours)}$ is compared with SPLT [140], Siam-R-CNN [126], SuperDiMP [62, 119], the tracker of Yu et al. [153], LTMU [137], STARK-ST50 [67], and KeepTrack [151]. It is worth noticing that the LTB-35 performance of many of such solutions shows a significant drop with respect to the LTB-50 one. This suggests that the additional sequences contained in the LTB-50 benchmark introduce challenging factors for such trackers. In contrast, our solution shows a limited performance decrease (-0.54%) indicating that it copes better with the issues introduced by the additional videos.

In Table 4.7 we compare our solution to SiamFC [43], MDNet [40], the tracker of Yue et al. [153], GlobalTrack [146], SuperDiMP, [62, 119] DMTrack [150], STARK-ST50 [67], and LTMU [137] on the TLP benchmark [143]. $\tau^{(ours)}$ results again the best tracker over both the considered metrics. Particularly, the Success_{TLP} improvement of the underlying trackers STARK-ST50 and SuperDiMP is of 6.9% and 9.3% respectively.

The LaSOT benchmark [38] comparison (presented in Table 4.8) of $\tau^{(ours)}$ with the trackers SuperDiMP [62, 119], TrDiMP [65], LTMU [137], Siam-R-CNN [126], TransT [66], STARK-ST50 [67], and KeepTrack [151], additionally confirms the effectiveness of the proposed solution for long-term visual object tracking scenarios.

4.5 Conclusions

In this chapter, we studied the problem of fusing the capabilities of complementary trackers in long-term tracking scenarios. To achieve such a goal, we designed an effective deep learning model to evaluate the tracking behavior of the underlying tracker STARK [67] and SuperDiMP [62, 119]. Based on such an evaluation, a decision strategy is implemented to select which of the two trackers is currently providing the best target localization. The outcome of such an operation is used to localize the target but also to correct the performance of the non-selected tracker. This strategy allows to ultimately correct the trackers from errors and drifts. We provided experimental results to understand the impact of the modules composing our solution. We also compared our methodology to the most recent schemes for long-term visual tracking on the LTB-35

[73], LTB-50 [133, 92], TLP [143], and LaSOT [38] benchmarks and achieved state-of-the-art results. Particularly, the instance of our proposed long-term tracking solution, referred to as mlpLT, resulted the winner of the Visual Object Tracking VOT2021 Long-Term Challenge [144].

5
Making Deep Learning Trackers Adapting to New Domains

Real-time visual object tracking is a key module in many application domains, and is especially used in robotic perception systems [154, 155, 156, 157, 158, 159]. Recently, the class of visual tracker based on deep regression [42, 41, 142] – also known as deep regression trackers (DRTs) – has been popularized in the robotics community [42] because of its efficiency and generality. Thanks to their simple architecture, DRTs achieve processing speeds that surpass 100 FPS, making them suitable even for low-resource robots. Moreover, with the availability of large-scale computer vision datasets [19], these trackers can learn to track a large variety of targets without relying on particular assumptions, thus simplifying the development of tracking pipelines. However, acquiring thousands of videos for training these systems is not realistic in many real-world robotic application domains. Additionally, many domains offer particular scenarios that differ much from the examples which DRTs are trained on. For example, drone [160] and driving [156, 161] applications require tracking objects from particular camera views. Underwater robots offer uncommon targets and settings [157, 162]. Other robotics systems can use different imaging modalities [155]. Robotic manipulation configurations need the tracking of atypical objects [163]. As shown in Figure 5.1, these situations cause DRTs' accuracy to be very low. This is due to their deep learning architecture that is subject to overfitting if trained directly on small application datasets, and suffers from the shift between training and test data distributions when trained for large-scale generic object tracking.

To address these issues, the visual tracking community proposes to increase the learning capacity of convolutional neural networks [45] (CNNs), or to use online learning mechanisms to adapt the capabilities of deep trackers [40, 61, 62] to every new target in every new video. These strategies lead to higher accuracy and robustness, but at cost of real-time speed achieved just on high-end machines. On the other hand, transfer learning [164] and domain adaptation [165] are widely used machine learning techniques to address such issues. The idea is to exploit the knowledge acquired in a source domain

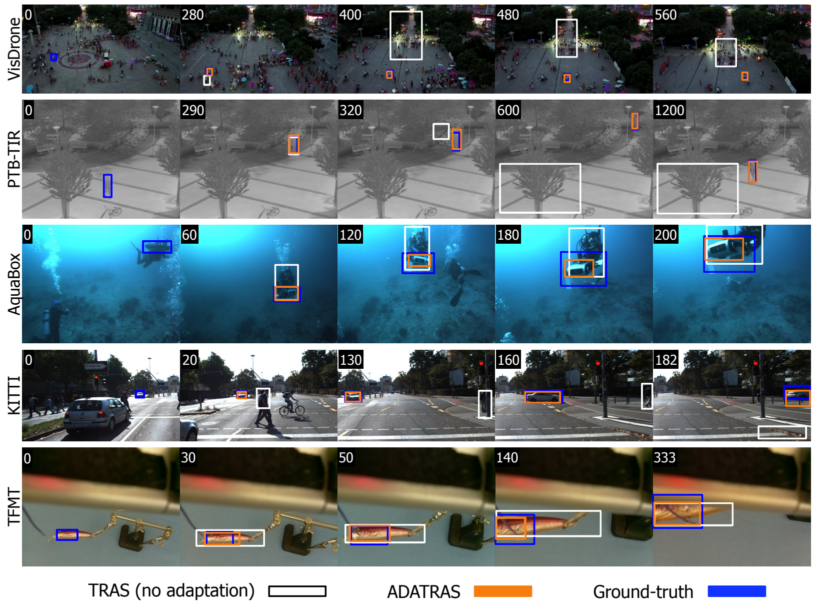


Figure 5.1: We propose a weakly-supervised strategy to adapt DRTs for particular and small-data robotic applications. This figure shows the performance of the TRAS tracker against its adapted version ADATRAS on five different robotic tracking settings. For better visualization, please check out the video at this link `https://youtu.be/3T3BJudDSwQ`. (© 2021 IEEE)

and adapt it to new target domains through an additional offline learning stage that exploits a few examples of the target domain. This allows to improve performance and generalization on the new domains, without sacrificing the test-time processing speed, as the deep models can be applied without any additional tuning. Different solutions have been proposed to adapt robotic vision systems [166, 167, 168], but no work considered adaptation in the context of robotic tracking.

Considering these motivations, the main contribution of this chapter is the first methodology for offline domain adaption of DRTs. This is also the first attempt in considering the domain adaptation problem in the context of visual tracking. To reduce the labeling effort and maintain application-specific development, we propose a weakly-supervised adaptation procedure. Thanks to reinforcement learning (RL), the knowledge previously acquired in a generic object tracking domain is adjusted with scalar signals that can be also delayed in time. But, as RL optimization is difficult, we build upon the experience of more accurate but slower trackers via knowledge distillation (KD) to stabilize learning and additionally improve the performance. We build our solution on the framework presented in Chapter 3, which marries KD and RL for generic object tracking. However, such a method is designed for learning DRTs with bounding-box level and densely annotated datasets. Hence, as an additional contribution, in this chapter we offer a generalization of the previously presented methodology that allows its exploitation in weakly labeled settings and for generic application objectives. Extensive analysis on five different robotic tracking domains shows that the proposed adaptation strategy makes DRTs applicable again on particular and low-resource robotic perception domains.

5.1 Related Work

5.1.1 Domain Adaptation in Robotic Vision

Domain adaptation has been previously studied in robotic vision. Spatial information about the domains has been exploited to adapt robotic vision system to recognize new objects [166]. Wulfmeier et al. [169] used adversarial approaches to adapt segmentation models to the visual appearance of weather and seasonal conditions. Batch normalization layers have been exploited for robotic vision-based kitting [170]. Particular loss functions [171, 167], augmentation networks [168], or pretext tasks [172] have been proposed extensively for the adaptation of visual capabilities from simulated to real environments. Bellocchio et al. [173] used generative adversarial networks to adapt robotic fruit counting systems to unseen species. Yet, no work considered the problem of adapting tracking knowledge acquired in a generic domain to another different robotic target domain. Furthermore, no method mixing RL and KD has been introduced in the context of domain adaptation.

5.1.2 Adaptation in Visual Tracking

In the visual tracking panorama, the concept of domain adaptation has been used to refer to instance-level online learning performed on the target object of every new test sequence. MDNet-based trackers [40] consider a training sequence as a domain and propose a CNN learned offline via binary classification on multiple domain-specific branches. Such a model is then refined on every test sequence by solving an online binary classification problem. ATOM [61] combines an offline learned bounding-box regression model with an efficient target-background IoU-based discriminator which is trained exclusively online. DiMP [62] performs an online update of a target-specific CNN model via a fast discriminative-learning optimization strategy. With respect to these works, our solution is conceptually different. We consider the target domain as a set of videos whose a subset is dedicated to offline learning. This introduces an additional training procedure, but it allows the adapted tracker to be extremely fast at test time, thanks to the avoidance of online adaptation mechanisms that reduce the tracking speed.

5.2 Methodology

We build our solution upon the framework presented in Chapter 3, which showed the effectiveness of combining KD and RL for generic object tracking. Differently from what described there, here we use RL to express a weak and temporally-delayed application-specific objective and employ KD to make the convergence achievable. We remark that our previous methodology is not applicable as it is for this new problem, because it assumes a fully supervised setting in which dense ground-truth bounding-box data is available.

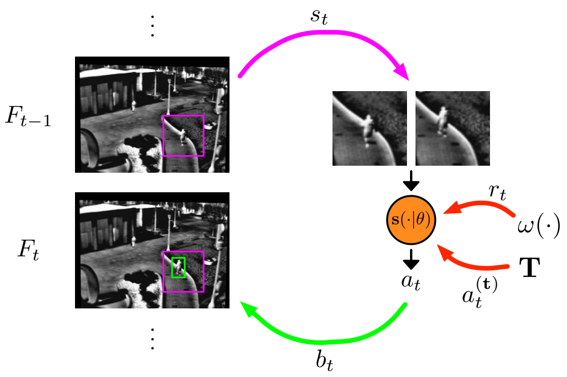


Figure 5.2: Visual representation of the MDP interaction process between the student and a video. At each step t, a state s_t is extracted from frames F_{t-1}, F_t. s_t is processed by $\mathbf{s}$ which outputs the action a_t that is transformed into the bounding-box output b_t. During the adaptation procedure, the learning is driven by the weak supervision function $\omega(\cdot)$ and by the actions of the set of teacher trackers $\mathbf{T}$. (© 2021 IEEE)

5.2.1 Preliminaries and Problem Statement

We consider a video $\mathcal{V}_j = \{F_t \in \mathcal{I}\}_{t=0}^{T_j}$ as a sequence of frames, where each F_t belongs to the space of RGB images $\mathcal{I} = \{0, \cdots, 255\}^{w \times h \times 3}$. Each video has a target object to be tracked, which is defined in the first frame F_0 through a bounding-box $b_0^{(g)} = [x_0^{(g)}, y_0^{(g)}, w_0^{(g)}, h_0^{(g)}] \in \mathbb{R}^4$ that specifies the coordinates of the object's top left corner, and its width and height. The goal of a tracker, given each frame F_t, is to predict the bounding-box $b_t = [x_t, y_t, w_t, h_t]$ that best fits the target in F_t.

As our solution is based on the KD framework, we employ the concepts of student and teacher [102]. We formally consider a regression-based tracker as the student $\mathbf{s} : \mathcal{I} \times \mathcal{I} \times \Theta \to \mathbb{R}^4$ which is a function parameterized by $\theta \in \Theta$ that outputs the relative motion of the target contained in two consecutive images given as input. We assume the student has acquired general tracking knowledge by optimizing θ on the videos of a source domain $\mathcal{D}^{(\text{source})}$. The set of teachers is defined as $\mathbf{T} = \{\mathbf{t} : \mathcal{I} \to \mathbb{R}^4\}$ where each $\mathbf{t}$ is a generic tracking function that, given a frame, produces a bounding-box for that frame.

Our problem of interest consists in adapting $\mathbf{s}(\cdot|\theta)$'s past ability to a new tracking domain $\mathcal{D}^{(\text{target})}$ different from $\mathcal{D}^{(\text{source})}$. More specifically, we assume $\mathcal{D}^{(\text{target})}$ is split into a training set $\mathcal{D}_{tr}^{(\text{target})} \subseteq \mathcal{D}^{(\text{target})}$, and a test set $\mathcal{D}_{te}^{(\text{target})} \subseteq \mathcal{D}^{(\text{target})}$ with $\mathcal{D}_{tr}^{(\text{target})} \cap \mathcal{D}_{te}^{(\text{target})} = \emptyset$. The goal is to exploit $\mathcal{D}_{tr}^{(\text{target})}$, for which weak supervision is given, to maximize the tracking performance on the videos of $\mathcal{D}_{te}^{(\text{target})}$.

5.2.2 Video Processing

To use RL, we model the tracking as an interaction process [55]. We treat $\mathbf{s}(\cdot|\theta)$ as an artificial agent which interacts with a video $\mathcal{V}_j \in \mathcal{D}^{(\text{target})}$ according to the Markov Decision Process (MDP) definition given in [142]. The interaction happens as a temporal

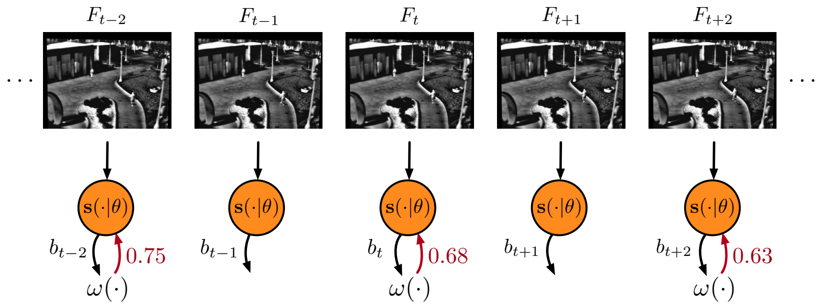


Figure 5.3: Visual representation of the student's feedback mechanism based on the weak supervision function $\omega(\cdot)$. At every t, $\mathbf{s}(\cdot|\theta)$ gives its bounding-box prediction b_t which is eventually evaluated with a 0-1 score by $\omega(\cdot)$ (if $\omega(\cdot)$ is defined for that temporal step). (© 2021 IEEE)

sequence of states $s_1, s_2, \cdots, s_t$, and actions $a_1, a_2, \cdots, a_t$. Every s_t is defined as a pair of image patches obtained by cropping F_{t-1} and F_t using the previously known bounding-box b_{t-1} and a factor c that enlarges the patches in order to include additional image context information. At the t-th step, the student is given the state s_t and outputs the continuous action a_t. Each a_t is defined as the vector $a_t = [\Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t] \in [-1, 1]^4$ which quantifies the relative horizontal and vertical translations ($\Delta x_t, \Delta y_t$, respectively) and width and height scale variations ($\Delta w_t, \Delta h_t$, respectively) that have to be applied to b_{t-1} to predict b_t. Hence, based on the previous bounding-box estimate [142], a_t is transformed into the bounding-box b_t which provides the localization of the target in frame F_t. A visual representation of the interaction procedure is depicted in Figure 5.2.

5.2.3 Weak Supervision

During adaptation on videos $\mathcal{V}_j \in \mathcal{D}_{tr}^{(\text{target})}$, the actions a_t of $\mathbf{s}$ are rewarded by the scalar value $r_t \in [-1, 1]$ (in RL terms, the reward). In our setting, this is what we use to express weak supervision. Differently from [142], who proposed a continuously available bounding-box overlap formulation, we just assume the feedback to be released as a 0-1 value through an arbitrary function $\omega : \mathbb{R}^4 \to [0, 1]$ that evaluates a bounding-box prediction b_t and that can be implemented based on the application needs. Additionally, we do not require $\omega(b_t)$ to be defined for every t. $\omega(b_t)$ is formally exploited in our proposed MDP reward definition which, at every t, is

$$r_t = r(b_t) = \begin{cases} 0 \text{ if } \omega(b_t) \text{ is not defined} \\ \nu\left(\omega(b_t)\right) \text{ if } \omega(b_t) \text{ is defined } \wedge \omega(b_t) \geq 0.5 \\ -1 \text{ otherwise} \end{cases} \tag{5.1}$$

with $\nu(z) = 2(\lfloor z \rfloor_{0.05}) - 1$ that floors to the closest 0.05 digit and shifts the input range from $[0, 1]$ to $[-1, 1]$. Figure 5.3 visualizes the proposed weak supervision mechanism.

5.2.4 Adapting the Tracker

The student's parameters θ, which have been pretrained on the $\mathcal{D}^{(\text{source})}$, are adapted to $\mathcal{D}^{(\text{target})}$ by learning offline on $\mathcal{D}_{tr}^{(\text{target})}$. To do this, we employ the end-to-end strategy proposed in [142] and we briefly report it by highlighting the improvements that allow its generalization for weak supervision.

Our adaptation strategy provides two learning objectives that are fulfilled at the same time. To optimize the actions with respect to $\omega(\cdot)$ (by means of the rewards), the following RL actor-critic loss formulation [58]

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\pi} + \mathcal{L}_{v} \tag{5.2}$$

$$\mathcal{L}_{\pi} = -\sum_{i=1}^{t_{max}} \log \mathbf{s}(s_i|\theta)\big(r_i + \gamma \mathbf{s}_v(s_{i+1}|\theta) - \mathbf{s}_v(s_i|\theta)\big) \tag{5.3}$$

$$\mathcal{L}_{v} = \sum_{i=1}^{t_{max}} \frac{1}{2}\big(R_i - \mathbf{s}_v(s_i|\theta)\big)^2, R_i = \sum_{k=1}^{i} \gamma^{k-1} r_k \tag{5.4}$$

is applied after t_{max} steps of interaction with $\mathcal{V}_j \in \mathcal{D}_{tr}^{(\text{target})}$, in which each a_t performed by $\mathbf{s}$ is sampled from a normal distribution $\mathcal{N}(\mu, \sigma)$. To attend this optimization goal, the student is set to produce the additional output $v_t = \mathbf{s}_v(s_t|\theta)$, which is the prediction of the γ-discounted cumulative reward R_i that $\mathbf{s}$ expects to receive from s_t to the end of the interaction. In RL terms, $\mathcal{L}_{\pi}$ and $\mathcal{L}_{v}$ are known as policy gradient loss with advantage and value loss respectively.

On a second side, our adaptation scheme minimizes the following objective

$$\mathcal{L}_{\text{dist}} = \sum_{i=1}^{t_{max}} |a_i^{(\mathbf{t})} - \mathbf{s}(s_i|\theta)| \cdot m_i, \tag{5.5}$$

which is the L1 loss [42, 41] between the actions performed by $\mathbf{s}$ and the actions $a_t^{(\mathbf{t})}$ that the teacher would take to move $\mathbf{s}$'s bounding-box b_{t-1} into the $\mathbf{t}$'s prediction $b_t^{(\mathbf{t})}$ [142]. Each of the differences in Eq. (5.5) are multiplied by the binary values m_i which represent the case in which $\mathbf{s}$ performs worse than the teacher. This learning objective makes the learning feasible and has the additional advantage of extracting knowledge from more accurate and robust tracking algorithms, leading ultimately to better performance. A distributed setting [174] is employed to implement the overall optimization strategy by considering $\frac{S}{2}$ students for the optimization of Eq. (5.2) and the other $\frac{S}{2}$ for Eq. (5.5).

The proposed adaptation procedure brings some modifications to the learning method of [142] that it allows to work in weakly supervised settings. First, the policy gradient $\log \mathbf{s}(s_i|\theta)$ term used in Eq. (5.3) is obtained after the definition of a normal distribution $\mathcal{N}(\mu, \sigma)$ with mean defined as $\mu = \mathbf{s}(s_t|\theta)$ and standard deviation σ considered with a fixed value. In this way, varying σ one can control $\mathbf{s}$'s exploration without the need of ground-truth bounding-boxes as in [142]. Second, we propose to favor $\mathbf{t}$'s prediction in the computation of $m_i \in \{0, 1\}$ of Eq. (5.5), by setting $m_i = 1$ if $r(b_t^{(\mathbf{t})}) \geq r(b_t)$ holds and $m_i = 0$ otherwise. This in order to address the 0 reward scenarios caused by the non definition of $\omega(\cdot)$, in which it is not possible to infer the actual student performance. Third, the $\mathbf{t}$ to which learn from in Eq. (5.5) is selected before the start of the interaction via some arbitrary decision strategy that can be defined depending on the application objectives and resources.

5.2.5 Tracking after Adaptation

After the adaptation-by-learning process is done, the student $\mathbf{s}(\cdot|\theta)$ is ready to be used for tracking on $\mathcal{D}_{te}^{(\text{target})}$ as follows. We consider each testing video $\mathcal{V}_j \in \mathcal{D}_{te}^{(\text{target})}$, for which the target is individuated in F_0 by the bounding-box $b_0^{(g)}$, as the aforementioned MDP. At each t, s_t are extracted from F_{t-1}, F_t, and a_t are performed by means of the student's adapted policy $\mathbf{s}(s_t|\theta)$ and transformed into bounding-box outputs b_t. We name the tracker resulting from this tracking procedure ADATRAS (ADApted TRAcking Student).

5.3 Experimental Setup

5.3.1 Performance Measures

To evaluate the performance of the trackers involved in this chapter, we follow the standard methodology introduced in [71]. Trackers are initialized in the first frame of a sequence with the target ground-truth bounding-box and let run until the end, respecting the one-pass evaluation (OPE) protocol. The quantitative measures used are the area under the curve (AUC) of the success and precision plots, which are referred as to success score (SS) and precision score (PS) respectively.

5.3.2 Tracker

We follow the most recent advancements in deep regression tracking [42, 175, 142] to implement our tracker $\mathbf{s}(\cdot|\theta)$ as a deep neural network with weights θ. The network gets as input s_t as two image patches which pass through two ResNet-18 [103] CNN branches with shared weights. The subsequent feature maps are linearized, concatenated together, and fed to two consecutive fully connected layers with ReLU activations and an LSTM layer [87], both with 512 neurons. The LSTM's output is finally fed to two fully connected heads that output the action $a_t = \mathbf{s}(s_t|\theta)$ and the state-value $v_t = \mathbf{s}_v(s_t|\theta)$ respectively.

5.3.3 Source and Target Domains

We conducted experiments considering the GOT-10k [39] and LaSOT [38] benchmarks as source domains $\mathcal{D}^{(\text{source})}$. These are large-scale tracking datasets containing, respectively, 10000 and 1400 videos of generic target objects (up to 563 to different object classes) in generic tracking settings. The initial optimization of θ on these sets was performed following the details of [142].

We demonstrate the capabilities of our solution on five robotic target domains, which we refer to as VisDrone, PTB-TIR, AquaBox, KITTI, and TFMT. These were selected

Table 5.1: Statistics of the target domains selected for this work. The number of training and test videos, frames, and the number of sequences after splitting the videos in chunks of 32 frames, are reported in the first five rows. SS and PS obtained by the teachers on the training videos are reported in the last three rows. (© 2021 IEEE)

Target Domain	VisDrone	PTB-TIR	AquaBox	KITTI	TFMT
$\lvert\mathcal{D}_{tr}^{(\text{target})}\rvert$	86	48	41	21	6
$\lvert\mathcal{D}_{te}^{(\text{target})}\rvert$	11	12	20	20	6
# frames $\mathcal{D}_{tr}^{(\text{target})}$	69941	23497	3927	4797	520
# frames $\mathcal{D}_{te}^{(\text{target})}$	7046	6532	6033	4143	1320
# splitted sequences $\mathcal{D}_{tr}^{(\text{target})}$	1696	804	263	356	42
$\mathbf{T}_{\texttt{M}}$ $\mathcal{D}_{tr}^{(\text{target})}$ SS PS	0.556 0.798	0.565 0.817	0.321 0.488	0.385 0.668	0.283 0.385
$\mathbf{T}_{\texttt{S}}$ $\mathcal{D}_{tr}^{(\text{target})}$ SS PS	0.576 0.741	0.617 0.785	0.499 0.717	0.430 0.579	0.460 0.659
$\mathbf{T}_{\texttt{A}}$ $\mathcal{D}_{tr}^{(\text{target})}$ SS PS	0.555 0.759	0.559 0.691	0.563 0.843	0.450 0.619	0.619 0.783

due to their particular characteristics in: camera views; uncommon objects and motions; image modality. Statistics of the domains are shown in Table 5.1. Beside being real-world robotic vision datasets, these domains also contain bounding-box labels for every frame of the videos. This permits an accurate validation with the control and simulation of the loss of supervision.

VisDrone. This domain concerns tracking objects in videos acquired from drones, and it is based on the publicly available VisDrone 2019 challenge dataset [176]. Targets available are persons, cars, or animals, acquired by particular camera views in which they appear very small and their motion is different depending on the drone's altitude. To implement $\mathcal{D}_{tr}^{(\text{target})}$ and $\mathcal{D}_{te}^{(\text{target})}$ we employed, respectively, the original training and validation sets provided by the authors [176].

PTB-TIR. The PTB-TIR target domain regards tracking people in videos acquired through a thermal-infrared (TIR) camera. This domain offers common objects (people), but video frames are represented via a different sensor. The data contained in the PTB-TIR benchmark [177] was employed. $\mathcal{D}_{tr}^{(\text{target})}$ and $\mathcal{D}_{te}^{(\text{target})}$ were obtained by randomly splitting, with an 80-20 ratio, the 60 videos contained in the benchmark.

AquaBox. This domain consists in tracking an underwater robot in an underwater video setting. Videos offer targets, camera views, motions, and object physics, that are very unusual from what available in standard tracking datasets. The data used was obtained with the AquaBox dataset [157]. For $\mathcal{D}_{tr}^{(\text{target})}$ and $\mathcal{D}_{te}^{(\text{target})}$ we employed the training and validation sets provided by the authors. Only videos composed of at least 25 frames were retained.

KITTI. This domain concerns tracking vehicles and people in videos acquired from a vehicle point of view, thus offering new camera views (different from the drone's) on common objects. We used the popular KITTI dataset [156] to implement $\mathcal{D}_{tr}^{(\text{target})}$ and $\mathcal{D}_{te}^{(\text{target})}$. We considered tracks longer than 100 frames of the KITTI's training videos and splitted them with a 50% ratio.

TFMT. This target domain requires tracking objects to perform a fine grained manipulation task with a robotic arm [163]. The targets and the settings contained in this dataset are very uncommon to generic object trackers. The 12 labeled videos contained in the TFMT dataset [163] have been splitted with a 50% ratio to implement $\mathcal{D}_{tr}^{(\text{target})}$ and $\mathcal{D}_{te}^{(\text{target})}$.

5.3.4 Weak Supervision Form

We experimented two forms of 0-1 function to weakly supervise $\mathbf{s}(\cdot|\theta)$. The first implements ω as the function $\omega^{(\text{iou})}(b_t, b_t^{(g)}) = \text{IoU}(b_t, b_t^{(g)})$, which takes the student's predicted bounding-box and a bounding-box reference $b_t^{(g)}$ of the target and computes their intersection-over-union as

$$\text{IoU}(b_t, b_t^{(g)}) = (b_t \cap b_t^{(g)})/(b_t \cup b_t^{(g)}). \tag{5.6}$$

The second form is through the 0-1 function $\omega^{(\text{dist})}(b_t, b_t^{(g)}) = 1 - \text{NormDist}(b_t, b_t^{(g)})$ where

$$\text{NormDist}(b_t, b_t^{(g)}) = \frac{\left\lceil \sqrt{(x_t - x_t^{(g)})^2 + (y_t - y_t^{(g)})^2} \right\rceil_{20}}{20} \tag{5.7}$$

is the function that computes the pixel distance between the centers of b_t and $b_t^{(g)}$, truncated at 20 and normalized by the same value. The value 20 was chosen following the standard precision score threshold defined in [71].

5.3.5 Teachers

The tracking teachers selected for this study are MDNet [40], SiamRPN++ [45], and ATOM [61]. Since they tackle visual tracking by different approaches, it is more likely that at least one can succeed in the application domain, and thus provide useful tracking knowledge. In the experiments, we considered exploiting single teachers or a pool of teachers. In particular, the following sets of teachers were examined $\mathbf{T}_{\texttt{M}} = \{\texttt{MDNet}\}, \mathbf{T}_{\texttt{S}} = \{\texttt{SiamRPN++}\}, \mathbf{T}_{\texttt{A}} = \{\texttt{ATOM}\}, \mathbf{T}_{\texttt{P}} = \{\texttt{MDNet}, \texttt{SiamRPN++}, \texttt{ATOM}\}$. The performance of these on $\mathcal{D}_{tr}^{(\text{target})}$ are shown in the last three rows of Table 5.1. We studied two methods to select the teacher $\mathbf{t}$ in Eq. (5.5). The first randomly selects $\mathbf{t} \in \mathbf{T}$ using a uniform distribution. The second selects $\mathbf{t}$ based on the average $\omega(\cdot)$ performance, that is

$$\mathbf{t} \in \mathbf{T} \, : \, \Omega(\mathbf{t}, \mathcal{V}_j) = \max_{\mathbf{t} \in \mathbf{T}} \Omega(\mathbf{t}, \mathcal{V}_j) \tag{5.8}$$

where

$$\Omega(\mathbf{t}, \mathcal{V}_j) = \frac{\sum_{t=0}^{T_j} \omega(b_t^{(\mathbf{t})})}{T_j} \tag{5.9}$$

is a 0-1 number that estimates the quality of $\mathbf{t}$'s predictions given for video $\mathcal{V}_j \in \mathcal{D}_{tr}^{(\text{target})}$.

Table 5.2: Comparison between ADATRAS and the teachers and DRTs. FPS are obtained on the SM machine. Best results, per domain and performance measure, are highlighted in red, second-best in blue, third-best in green. (© 2021 IEEE)

Tracker	VisDrone			PTB-TIR			AquaBox			KITTI			TFMT		
	SS	PS	FPS	SS	PS	FPS	SS	PS	FPS	SS	PS	FPS	SS	PS	FPS
MDNet [40]	**0.559**	**0.902**	2	0.586	**0.953**	2	0.543	0.504	2	0.413	**0.686**	3	**0.501**	**0.770**	2
SiamRPN++ [45]	0.532	0.790	31	**0.614**	0.774	55	**0.591**	**0.753**	41	**0.504**	**0.632**	43	0.504	0.637	27
ATOM [61]	**0.539**	**0.891**	16	**0.620**	**0.778**	24	**0.594**	**0.742**	20	**0.529**	**0.686**	19	**0.615**	**0.639**	28
GOTURN [41]	0.350	0.600	56	0.327	0.497	**200**	0.402	0.468	**110**	0.209	0.313	62	0.363	0.271	**125**
RE3 [42]	0.354	0.626	**60**	0.201	0.278	**255**	0.445	0.398	107	0.221	0.308	**63**	0.406	0.272	**136**
TRAS [142]	0.384	0.653	**65**	0.432	0.603	168	0.522	0.619	**161**	0.486	0.627	**85**	0.332	0.169	112
ADATRAS	**0.552**	**0.823**	**67**	**0.661**	**0.862**	**170**	**0.576**	**0.732**	**165**	**0.537**	**0.720**	**89**	**0.581**	**0.659**	**115**

5.3.6 Implementation Details

To produce more training samples, each video (and the respective filtered bounding-box sequences) was split in 20 randomly indexed sequences of 32 frames, following [42, 142]. The total number of videos is reported in row five of Table 5.1. A temporal reverse of the sequences was also applied with 50% probability during training. $S = 12$ training students were used for training. $\sigma = 0.05$ was set for the VisDrone PTB-TIR, KITTI, and TFMT domains, and $\sigma = 0.025$ for AquaBox. To facilitate the learning, a curriculum learning procedure similar to [42] was employed by increasing the length of the interaction during the adaptation procedure. The Adam optimizer [90] was utilized. A learning rate of $7.5 \cdot 10^{-7}$ was set for all the layers of the student except for the fully connected layer that predicts v_t, for which learning rate was set to 10^{-5}. The student was trained until the validation performance stopped improving. Trainings took between 12 and 48 hours, depending on the amount of data available in a domain. We experienced some variance between the outcomes of different experiment runs, and therefore we report averaged results. Other settings not specified in this section have been inherited from [142]. In the following of the chapter, if not specified otherwise, default experimental settings are with the student initially optimized on GOT-10k-based $\mathcal{D}^{(\text{source})}$, learning from $\mathbf{T}_{\text{P}}$ by respecting Eq. (5.8), with weak supervision based on $\omega^{(\text{iou})}(\cdot)$ given at every time step, and with the hyper-parameters mentioned above.

5.3.7 Hardware and Software

We employed three hardware machines for our experiments. A high-end server machine with an Intel Xeon E5-2690 v4 @ 2.60GHz CPU, 320 GB of RAM, and 4 NVIDIA TITAN V GPUs. We refer to this as SM. A desktop computer with an Intel Xeon W-2125 @ 4.00GHz CPU, 32GB of RAM, and an NVIDIA GTX1080-Ti, which we refer to as DM. And an embedded board NVIDIA Jetson Nano with an ARM A57 quad-core 1.43 GHz CPU, 4GB of RAM, and a Maxwell 128 core GPU, which we refer to as ED. All the code was implemented in Python. Trainings were performed on the SM machine.

5.4 Results

5.4.1 Comparison with other Trackers

Table 5.2 reports the performance of ADATRAS in comparison with teachers and other DRTs on the considered robotic domains. After adaptation, our proposed tracker com-

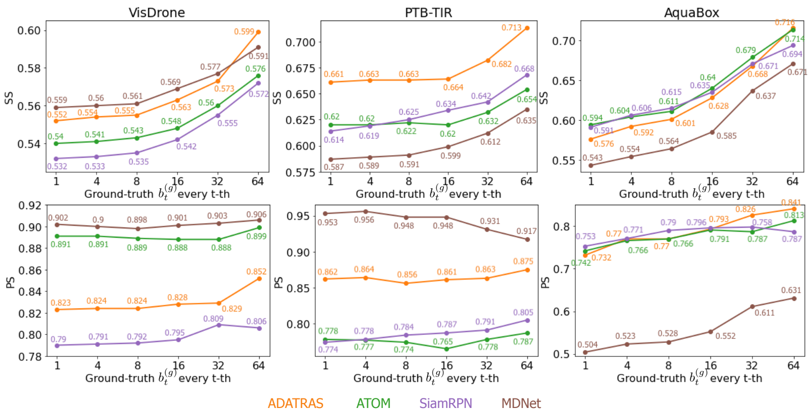


Figure 5.4: Effect of evaluating the trackers with a weakly-labeled test set in which ground-truths are available every 4-th, 8-th, 16-th, 32-th, and 64-th frame. (© 2021 IEEE)

Table 5.3: Speed performance in FPS of ADATRAS and the teachers on different machines. Best results, per machine, are highlighted in bold. (ATOM w/o GPU results were not obtained because the implementation was not designed to run without it.) (© 2021 IEEE)

Tracker		VisDrone			PTB-TIR			AquaBox		
		SM	DM	ED	SM	DM	ED	SM	DM	ED
MDNet [40]	w GPU	2	3	< 1	2	3	< 1	2	3	< 1
	w/o GPU	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
SiamRPN++ [45]	w GPU	31	28	1	55	43	1	41	35	1
	w/o GPU	3	3	< 1	3	3	< 1	3	3	< 1
ATOM [61]	w GPU	16	28	3	24	59	4	20	43	4
	w/o GPU	-	-	-	-	-	-	-	-	-
ADATRAS	w GPU	**67**	**80**	**15**	**170**	**216**	**23**	**165**	**185**	**26**
	w/o GPU	**33**	**55**	**2**	**62**	**91**	**2**	**98**	**168**	**2**

petes with the top-performing trackers generally. On some domains (e.g.PTB-TIR, KITTI) it also outperforms them. Regarding the processing speed, ADATRAS is always faster than these methods. The performance of TRAS [142], RE3 [42] and GOTURN [41] demonstrate the difficulties of DRTs due to the domain shift. In Figure 5.1, qualitative results of ADATRAS in comparison with the non adapted TRAS are presented. Given these results, our methodology allows to make DRTs accurate as state-of-the-art visual trackers in challenging robotic vision domains.

We analyse in depth our methodology on the VisDrone, PTB-TIR, AquaBox domains from now on. Figure 5.4 shows how the performance of ADATRAS and the teachers change considering weak supervision also for $\mathcal{D}_{te}^{(\text{target})}$. In particular, the SS and PS performance was analyzed with ground-truths $b_t^{(g)}$ available every 4-th, 8-th, 16-th, 32-th, and 64-th frame. Overall, the performance tends to increase as fewer references are used for evaluation, especially for domains with a smaller number of frames. For some less-frequent $b_t^{(g)}$ settings, ADATRAS results more accurate than the teachers.

Table 5.4: Comparison of the proposed weakly-supervised domain adaptation method (last row) with baselines that: do not adapt; are trained from scratch; do fine-tuning. Best results, per method, are highlighted in bold. (© 2021 IEEE)

Method	VisDrone		PTB-TIR		AquaBox	
	SS	PS	SS	PS	SS	PS
no adaptation (TRAS)	0.384	0.653	0.432	0.603	0.522	0.619
from scratch by [142]	0.459	0.712	0.585	0.751	0.601	0.670
fine-tuning by [142]	0.549	0.795	0.659	0.848	0.569	**0.762**
from scratch by proposed	0.475	0.722	0.625	0.819	0.549	0.695
adaptation by proposed	**0.552**	**0.823**	**0.661**	**0.862**	**0.576**	0.732

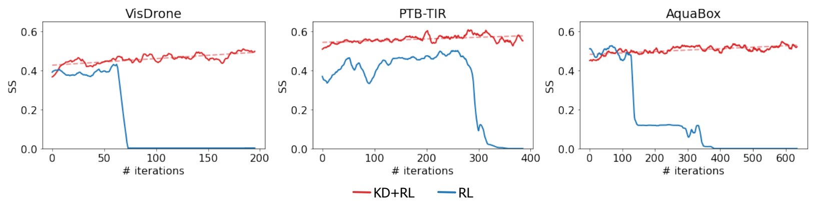


Figure 5.5: SS trends on $\mathcal{D}_{te}^{(\text{target})}$ of the proposed adaptation strategy (red line) and a pure RL fine-tuning (blue line) at different iterations during the learning phase. After some iterations, the RL-based solution diverges, while the proposed adaptation continuously improves ADATRAS (as shown by the red dashed line). Lines have been smoothed for better visualization. (© 2021 IEEE)

5.4.2 Speed Analysis

Table 5.3 reports the analysis on the processing speed of our method in comparison with the teachers on different machines. ADATRAS results the fastest method on all machine setups. Very high speeds are reached on top machines with a GPU (SM or DM), leaving large space for real-time downstream application development and making it good for lower resource robots. Remarkably, ADATRAS achieves real-time speed even without a GPU when run on top machines. When run on small embedded devices like ED, ADATRAS achieves real-time speeds on PTB-TIR and AquaBox and a quasi-real-time speed on VisDrone, considering 20 FPS as real-time baseline [92, 119].

5.4.3 Comparison with Baselines

Table 5.4 presents the performance of our methodology in comparison with baseline adaptation and no-adaptation methods. Our method (in the last row) outperforms the tracker without adaptation (TRAS [142]) on every target domain. These results show that the goal of improving the baseline tracker's accuracy with weak supervision is achieved. Generally, performance is even improved with respect to the dense bounding-box supervision experiments performed following [142], thus justifying the introduced improvements. Adapting past knowledge is effective to reduce overfitting, as demonstrated by the improvement over the results reported in rows two and four, for which training was performed from scratch. Figure 5.5 shows how the adaptation procedure with only an RL signal causes the student to diverge after some iterations. This is prob-

Table 5.5: Performance comparison between ADATRAS and trackers fine-tuned on $\mathcal{D}_{tr}^{(\text{target})}$. Best results, per tracker, are highlighted in bold. (© 2021 IEEE)

Tracker	VisDrone		PTB-TIR		AquaBox	
	SS	PS	SS	PS	SS	PS
SiamRPN++-ft [45]	**0.594**	**0.842**	0.572	0.733	0.559	0.703
GOTURN-ft [41]	0.380	0.642	0.353	0.548	0.450	0.374
RE3-ft [42]	0.168	0.297	0.408	0.554	0.530	0.586
ADATRAS	0.552	0.823	**0.661**	**0.862**	**0.576**	**0.732**

Table 5.6: Performance of the proposed tracker under different supervision settings. Best results, per supervision method, are highlighted in bold. (© 2021 IEEE)

Supervision	VisDrone		PTB-TIR		AquaBox	
	SS	PS	SS	PS	SS	PS
GT $b_t^{(g)}$	0.490	0.757	0.640	0.803	0.517	0.629
KD $b_t^{(\mathbf{t})}$	0.497	0.769	0.601	0.788	0.514	0.634
$\omega^{(\text{iou})}(\cdot)$	**0.552**	0.823	**0.661**	0.862	**0.576**	0.732
$\omega^{(\text{dist})}(\cdot)$	0.523	**0.852**	0.638	**0.894**	0.575	**0.774**

ably due to the increased length of the interaction (based on the curriculum strategy) that causes wrong gradient estimations. Table 5.5 shows the performance of ADATRAS in comparison with other trackers fine-tuned (following their original learning strategy) on $\mathcal{D}_{tr}^{(\text{target})}$. Our approach results generally better. This can be attributed to the RL strategy, which leads to more efficient data exploration, ultimately providing a data augmentation effect.

5.4.4 Weak Supervision Analysis

Table 5.6 reports the performances of ADATRAS adapted with different kind of supervision. The functions $\omega^{(\text{iou})}(\cdot), \omega^{(\text{dist})}(\cdot)$ improve by a good margin the results achieved by learning from ground-truth bounding-boxes (GT $b_t^{(g)}$), or by learning just from the bounding-box predictions given by the teachers (KD $b_t^{(\mathbf{t})}$). Using $\omega^{(\text{iou})}(\cdot)$ allows to achieve the best results in SS, while using $\omega^{(\text{dist})}(\cdot)$ improves the PS performance. These results confirm that optimizing a specific performance measure as reward function induces the improvement of such measure at test time. Moreover, we analyzed the sensibility of ADATRAS to the weak supervision delayed in time. In particular, ADATRAS was trained with weak supervision happening every 4-th, 8-th, 16-th, 32-th, 64-th, temporal step t (in a 20 FPS setting, this would mean every 0.2, 0.4, 0.8, 1.6, 3.2, seconds). Results are shown in Figure 5.6. In general, performance tends to decrease as the supervision is more delayed, but without a significant loss, especially for domains with a larger number of frames (e.g. VisDrone). Interestingly, particularly delayed supervision allows achieving similar performance as the case in which supervision is given more frequently. We hypothesize this is due to the distribution between supervised $\mathcal{D}_{tr}^{(\text{target})}$'s frames and the frames appearing in $\mathcal{D}_{te}^{(\text{target})}$. More importantly, our proposed adaptation strategy reaches and surpasses the GT $b_t^{(g)}$ adaptation (row one of Table 5.4) even with delayed supervision.

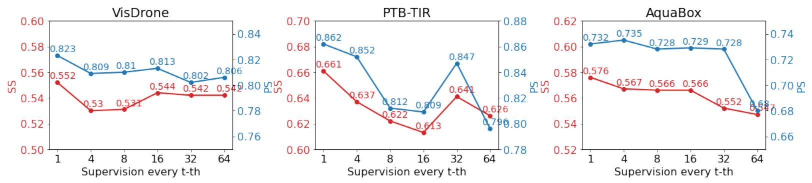


Figure 5.6: Performance of ADATRAS trained on the three domains considering weak supervision delayed at different steps in time. (© 2021 IEEE)

Table 5.7: Performance comparison of ADATRAS adapted starting from different source domain knowledge. Performance without adaptation is also reported in the first two rows. Best results are highlighted in bold. (© 2021 IEEE)

Source	VisDrone		PTB-TIR		AquaBox	
	SS	PS	SS	PS	SS	PS
no-adaptation - LaSOT	0.233	0.475	0.359	0.591	0.328	0.335
no-adaptation - GOT-10k	0.384	0.653	0.432	0.603	0.522	0.619
adaptation - LaSOT	0.475	0.726	0.646	0.860	0.541	0.719
adaptation - GOT-10k	**0.552**	**0.823**	**0.661**	**0.862**	**0.576**	**0.732**

5.4.5 Impact of Source and Teachers

Table 5.7 reports the performance of ADATRAS which $\mathbf{s}(\cdot|\theta)$ is adapted after learning on two different source domains. The performance of the two settings before adaptation are reported in the first two rows. $\mathbf{s}(\cdot|\theta)$ trained on the GOT-10k dataset performs much better than $\mathbf{s}(\cdot|\theta)$ optimized on the LaSOT dataset, due to the broader knowledge acquired on the larger GOT-10k. The results in rows three and four show that such a trend is maintained also after adaptation, showing that a better baseline tracking behavior is an important factor to achieve a better adapted tracking policy.

In Table 5.8 ADATRAS was analyzed after adaptation with different teacher setups. Using a better teacher does not translate into better performance, suggesting that it is important to understand on which sequences teachers perform well. Best results are obtained by learning from multiple teachers, demonstrating that the knowledge of these is compensated on $\mathcal{D}_{tr}^{(\text{target})}$. Moreover, giving a ranking of teachers and selecting the best for each training video, allows better performance on domains where teachers are better and more training frames are available (e.g. VisDrone and PTB-TIR). For some domain, selecting them randomly leads to almost equal performances than the ranking-based selection, probably due to the data distributions of $\mathcal{D}_{tr}^{(\text{target})}$ and $\mathcal{D}_{te}^{(\text{target})}$ which don't reflect the average performance of the teachers.

5.5 Conclusions

In this chapter, we studied the problem of adapting deep learning-based trackers to new and particular domains in the context of robotic vision. In particular, we presented the first methodology for domain adaption of DRTs, which are fast but inaccurate visual trackers popular in the robotics community. We achieved our goal by proposing a weakly-supervised adaptation approach, thus reducing the labeling effort. RL has been

Table 5.8: Performance of the proposed tracker with different teacher setups. Best results, per setup, are highlighted in bold. (© 2021 IEEE)

Teachers	VisDrone		PTB-TIR		AquaBox	
	SS	PS	SS	PS	SS	PS
$\mathbf{T}_{\mathtt{M}}$	0.525	0.786	0.542	0.733	0.521	0.556
$\mathbf{T}_{\mathtt{S}}$	0.467	0.731	0.576	0.774	0.582	0.642
$\mathbf{T}_{\mathtt{A}}$	0.466	0.734	0.604	0.755	0.537	0.729
$\mathbf{T}_{\mathtt{P}}$ random selection	0.533	0.791	0.654	0.825	**0.603**	**0.735**
$\mathbf{T}_{\mathtt{P}}$ with Eq. (5.8)	**0.552**	**0.823**	**0.661**	**0.862**	0.576	0.732

used to express weak supervision as a scalar application-dependent and temporally-delayed feedback. KD was employed to guarantee convergence of the deep learning models, and to transfer knowledge from other trackers. Extensive experiments on five different domains and three machine setups demonstrated the effective usage of our methodology for various robotic perception domains. Real-time speed was achieved on small embedded devices and on machines without GPUs. Accuracy was comparable to more powerful but slow state-of-the-art trackers.

6 First Person Vision: A New Challenging Domain

Chapter 5 introduced the problem of domain adaptation in the context of visual tracking and demonstrated how particular characteristics of the deployment scenario make deep learning trackers lose their accuracy. In this chapter, we focus on a domain that poses major challenges not only to deep learning-based trackers but also to trackers based on old-fashioned methods. This domain is First Person Vision (FPV) [178]. This is a sub-field of computer vision devoted to the study of algorithms applied on images and videos acquired from a camera mounted on the head of a person (the latter is usually referred as the camera viewer). The point of view captured with such a setup is the most similar to the one naturally arising in human beings. In simpler terms, FPV can be considered as the area of computer science that aims to make computers see as the humans do through their eyes. Systems that employ such a camera arrangement are exploited in all those tasks that require the understanding of the interactions happening between a camera wearer and the surrounding objects [179, 180, 181, 182, 183, 184, 185, 186, 187, 188]. Obtaining such information is fundamental because it could be used to develop high-level technologies to assist the people wearing cameras. For example, a worker in a factory could wear a camera system capable of recognize his/her actions, understand his/her intentions and eventually alert him/her before an accident would occur. Furthermore, building effective computer vision system in first person videos could lead to the development of robots better integrated with people.

But to discover how interactions between the person and objects are progressing, the continuous knowledge of where objects of interest are located inside the video frame is necessary. Indeed, keeping track of object locations over time allows to understand which objects are moving, which of them are passively captured while not interacted, and how the user relates to the scene.

The benefits of tracking in FPV have been explored by a few previous works in the literature. For example, visual trackers have been exploited in solutions to comprehend social interactions through faces [189, 190, 191], to improve the performance of

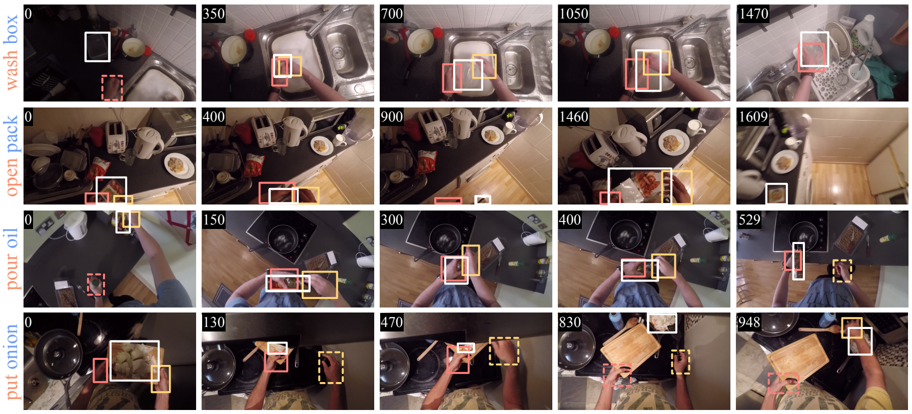


Figure 6.1: In this paper, we study the problem of visual object tracking in the context of FPV. To achieve such a goal, we introduce a new benchmark dataset named TREK-150, of which some qualitative examples of sequences are represented in this Figure. In each frame, the white rectangle represents the ground-truth bounding box of the target object. The orange and yellow boxes localize left and right hands respectively (plain lines indicate the interaction between the hand and the target). Each number in the top left corner reports the frame index. For each sequence, the action performed by the camera wearer is also reported (verb in orange, noun in blue). As can be noted, objects undergo significant appearance and state changes due to the manipulation by the camera wearer, which makes the proposed setting challenging for current trackers.

hand detection for rehabilitation purposes [192], to capture hand movements for action recognition [193], and to forecast human-object interactions through the analysis of hand trajectories [181]. Such applications have been made possible trough the development of customized tracking approaches to track specific target categories like people [194, 195], people faces [189, 191], or hands [193, 181, 192, 196, 197, 198] from a first person perspective.

Despite the aforementioned attempts to leverage tracking in egocentric vision pipelines, the standard approach to generic-object continuous localisation in FPV tasks still relies on detection models that evaluate video frames independently [180, 199, 200, 182, 201, 202, 179, 203]. This paradigm has the drawback of ignoring all the temporal information coming from the object appearance and motion contained in consecutive video frames. Also, it generally requires a higher computational cost due to the need to repeat the detection process in every frame. In contrast, visual object tracking aims to exploit past information about the target to infer its position and shape in the next frames of a video [4, 5]. This process can improve the efficiency of algorithmic pipelines because of the reduced computational resources needed, but most importantly because it allows to maintain the spatial and temporal reference to specific object instances.

Visually tracking a generic object in an automatic way introduces several different challenges that include occlusions, pose or scale changes, appearance variations, and fast motion. The computer vision community has made significant progress in the development of algorithms capable of tracking arbitrary objects in unconstrained scenarios affected by those issues. The advancements have been possible thanks to the development of new and effective tracking principles [31, 32, 43, 60, 62, 136, 98, 137, 67], and to the careful design of benchmark datasets [71, 70, 135, 204, 38, 39] and competitions [83, 84, 92, 119, 144] that well represent the aforementioned challenging situations. However, all these research endeavours have taken into account mainly the classic third person scenario in which the target objects are passively observed from an external point of view and where they do not interact with the camera wearer. It is a matter of fact that the nature of images and videos acquired from the first person viewpoint is inherently different from the type of image captured from video cameras set as on an external point of view. As we will show in this paper, the particular characteristics of FPV, such as the interaction between the camera wearer and the objects as well as the proximity of the scene and the camera's point of view, cause the aforementioned challenges to occur with a different nature and distribution, resulting in the persistent occlusion, significant scale and state changes of objects, as well as an increased presence of motion blur and fast motion (see Figure 6.1).

While the use cases of object tracking in egocentric vision are manifold and the benefit of tracking generic objects is clear as previously discussed, it is evident that visual object tracking is still not a dominant technology in FPV. Only very recent FPV pipelines are starting to employ generic object trackers [205, 191], but a solution specifically designed to track generic objects in first person videos is still missing. We think this lack of interest towards visual object tracking in FPV is mainly due to the limited amount of knowledge present in the literature about the capabilities of current visual object trackers in FPV videos. Indeed, this gap in the research opens many questions about the impact of the first person viewpoint on visual trackers: can the trackers available nowadays be used "off-the-shelf"? How does FPV impact current methodologies? Which tracking approaches work better in FPV scenarios? What factors influence the most the tracking performance? What is the contribution of trackers in FPV? We believe that the particular setting offered by FPV deserves a dedicated analysis that is still missing in the literature, and we argue that further research on this problem cannot be pursued without a thorough study on the impact of FPV on tracking.

In this chapter, extensively analyze the problem of visual object tracking in the FPV domain in order to answer the aforementioned questions. Given the lack of suitable benchmarks, we follow the standard practice of the visual tracking community that suggests to build a curated dataset for evaluation [71, 134, 70, 204, 135, 92, 206]. Hence, we propose a novel visual tracking benchmark, TREK-150 (TRacking-Epic-Kitchens-150), which is obtained from the large and challenging FPV dataset EPIC-KITCHENS (EK) [179, 203]. TREK-150 provides 150 video sequences which we densely annotated with the bounding boxes of a single target object the camera wearer interacts with. The dense localization of the person's hands and the interaction state between those and the target are also provided. Additionally, each sequence has been labeled with attributes that identify the visual changes the object is undergoing, the class of the target object, as well as the action he/she is performing. By exploiting the dataset, we present an extensive and in-depth study of the accuracy and speed performance of 38 established generic object trackers and of 2 newly introduced baseline FPV trackers. We leverage standard evaluation protocols and metrics and propose new ones. This is done in order to evaluate the capabilities of the trackers in relation to specific FPV scenarios. Furthermore, we assess the trackers' performance by evaluating their impact on the FPV-specific downstream task of human-object interaction detection.

In sum, the main contribution of this Chapter is the first systematic analysis of visual object tracking in FPV. In addition to that, our study brings additional innovations:

(i) the description and release of the new TREK-150 dataset, which offers new challenges and complementary features with respect to existing visual tracking benchmarks;

(ii) a new measure to assess the tracker's ability to maintain temporal reference to targets;

(iii) a protocol to evaluate the performance of trackers with respect to a downstream task;

(iv) two FPV baseline trackers combining a state-of-the-art generic object tracker and FPV object detectors.

Our results show that FPV offers new and challenging tracking scenarios for the most recent and accurate trackers [137, 65, 61, 207, 60] and even for FPV trackers. We study the factors causing such performance and highlight possible future research directions. Despite the difficulties introduced by FPV, we prove that trackers bring benefits to FPV downstream applications requiring short-term object tracking such as hand-object interaction. Given our results and considering the potential impact in FPV, we expect that generic object tracking will gain popularity in this domain as new and FPV-specific methodologies are investigated.[1]

6.1 Related Work

6.1.1 Visual Tracking in FPV

There have been some attempts to tackle visual tracking in FPV. Alletto et al. [194] improved the TLD tracker [138] with a 3D odometry-based module to track people. For a similar task, Nigam et al. [195] proposed EgoTracker, a combination of the Struck [208] and MEEM [99] trackers with a person re-identification module. Face tracking was tackled by Aghaei et al. [189] through a multi-object tracking approach termed extended-bag-of-tracklets. Hand tracking was studied in several works [193, 192, 196, 197, 198]. Sun et al. [198] developed a particle filter framework for hand pose tracking. Mueller et al. [196] instead proposed a solution based on an RGB camera and a depth sensor, while Kapidis et al. [193] and Visée et al. [192] combined the YOLO [209] detector trained for hand detection with a visual tracker. The former work used the multi-object tracker DeepSORT [210], whereas the latter employed the KCF [32] single object tracker. Han et al. [197] exploited a detection-by-tracking approach on video frames acquired with 4 fisheye cameras.

All the aforementioned solutions focused on tracking specific targets (i.e., people, faces, or hands), and thus they are likely to fail in generalizing to arbitrary target

[1]Annotations, trackers' results, and code are available at `https://machinelearning.uniud.it/datasets/trek150/`.

Table 6.1: Statistics of the proposed TREK-150 benchmark compared with other benchmarks designed for SOT evaluation.

Benchmark	OTB-50 [215]	OTB-100 [71]	TC-128 [134]	UAV123 [70]	NUS-PRO [204]	NfS [135]	VOT2019 [92]	CDTB [206]	TREK-150
# videos	51	100	128	123	365	100	60	80	150
# frames	29K	59K	55K	113K	135K	383K	20K	102K	97K
Min frames across videos	71	71	71	109	146	169	41	406	161
Mean frames across videos	578	590	429	915	371	3830	332	1274	649
Median frames across videos	392	393	365	882	300	2448	258	1179	484
Max frames across videos	3872	3872	3872	3085	5040	20665	1500	2501	4640
Frame rate	30 FPS	30 FPS	30 FPS	30 FPS	30 FPS	240 FPS	30 FPS	30 FPS	60 FPS
# target object classes	10	16	27	9	8	17	30	23	34
# sequence attributes	11	11	11	12	12	9	6	13	17
FPV	✗	✗	✗	✗	✗	✗	✗	✗	✓
# action verbs	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	20

objects. Moreover, they have been validated on custom designed datasets, which limits the reproducibility of the works and the ability to compare them to other solutions. In contrast, we focus on the evaluation of algorithms for the generic object tracking task. We design our evaluation to be reproducible and extendable by releasing TREK-150, a dataset of 150 videos of different objects manipulated by the camera wearer, which we believe will be useful to study object tracking in FPV. To the best of our knowledge, ours is the first attempt to evaluate systematically and in-depth generic object tracking in FPV.

6.1.2 Visual Tracking for Generic Settings

In recent years, there has been an increased interest in developing accurate and robust tracking algorithms for generic objects and domains. Preliminary trackers were based on mean shift algorithms [25], key-point [28], part-based methods [29, 30], or SVM learning [208]. Later, solutions based on correlation filters gained popularity thanks to their processing speed [31, 32, 33, 34, 211]. More recently, algorithms based on deep learning have been proposed to extract efficient image and object features. This kind of representation has been used in deep regression networks [41, 42], online tracking-by-detection methods [40, 207], approaches based on reinforcement learning [50, 175, 212, 117], deep discriminative correlation filters [60, 61, 62, 127, 124, 130], and trackers based on siamese networks [43, 45, 47, 213, 136, 98, 214]. All these methods have been designed for tracking arbitrary target objects in unconstrained domains. However, no solution has been studied and validated on a number of diverse FPV sequences as we propose in this study.

6.1.3 Visual Tracking Benchmarks

Disparate bounding-box level benchmarks are available today to evaluate the performance of single-object visual tracking algorithms. The Object Tracking Benchmarks (OTB) OTB-50 [215] and OTB-100 [71] are two of the most popular benchmarks in the visual tracking community. They provide 51 and 100 sequences respectively including generic target objects like vehicles, people, faces, toys, characters, etc. The Temple-Color 128 (TC-128) dataset [134] comprises 128 videos that were designed for the evaluation of color-enhanced trackers. The UAV123 dataset [70] was constructed to benchmark the tracking of 9 classes of target in 123 videos captured by unmanned aerial vehicles (UAVs) cameras. The NUS-PRO dataset [204] contains 365 sequences and aims to benchmark human and rigid object tracking with targets belonging to one of 8 categories. The Need for Speed (NfS) dataset [135] provides 100 sequences with a frame rate of 240 FPS. The aim of the authors was to benchmark the effects of frame rate variations on the tracking performance. The VOT2019 benchmark [92] was the last iteration of the annual Visual Object Tracking challenge that required bounding-boxes as target object representation. This dataset contains 60 highly challenging videos, with generic target objects belonging to 30 different categories. The Color and Depth Tracking Benchmark (CDTB) dataset [206] offers 80 RGB sequences paired with a depth channel. This benchmark aims to explore the use of depth information to improve tracking. Following the increased development of deep learning-based trackers, large-scale generic-domain tracking datasets have been recently released [37, 39, 38]. These include more than a thousand videos normally split into training and test subsets. The evaluation protocol associated with these sets requires the evaluation of the trackers after they have been trained on the provided training set.

Even though all the presented benchmarks offer various tracking scenarios, limited work has been done in the context of FPV, with some studies tackling the problem of tracking pedestrians or cars from a moving camera [216]. Some datasets of egocentric videos such as ADL [217] and EK-55 [179] contain bounding-box object annotations. However, due to the sparse nature of such annotations (typically 1/2 FPS), these datasets cannot be used for the accurate evaluation of trackers in FPV. To the best of our knowledge, TREK-150 is the first benchmark for tracking objects which are relevant to (or manipulated by) a camera wearer in egocentric videos. We believe that TREK-150 is tantalizing for the tracking community because it offers complementary tracking situations and new target object categories that are not present in other tracking benchmarks. Since in this study we aim to benchmark generic approaches to visual tracking (that would not necessarily consider the deep learning approach), we follow the practice of previous works [71, 134, 70, 204, 135, 92, 206] and set up a well described dataset for evaluation of generic visual trackers. We believe that TREK-150 can be a useful research tool from both the FPV and visual tracking research communities.

6.2 The TREK-150 Benchmark

In this section, we describe TREK-150, the novel dataset proposed for the study of the visual object tracking task in FPV. TREK-150 is composed of 150 video sequences. In each sequence, a single target object is labeled with a bounding box which encloses the appearance of the object in each frame in which the object is visible (as a whole or in part). Every sequence is additionally labeled with attributes describing the visual variability of the target and the scene in the sequence. To study the performance of trackers in the setting of human-object interaction, we provide bounding box localization of hands and labels for their state of interaction with the target object. Moreover, two additional verb and noun attributes are provided to indicate the action performed by the person and the class of the target, respectively. Some qualitative examples of the video sequences with the relative annotations are shown in Figure 6.1. Table 6.1 reports key statistics of our dataset in comparison with existing tracker evaluation benchmarks. It

Table 6.2: Selected sequence attributes. The first block of rows describes attributes commonly used by the visual tracking community. The last four rows describe additional attributes introduced in this paper to characterize FPV tracking sequences.

Attribute	Meaning
SC	Scale Change: the ratio of the bounding-box area of the first and the current frame is outside the range [0.5, 2]
ARC	Aspect Ratio Change: the ratio of the bounding-box aspect ratio of the first and the current frame is outside the range [0.5, 2]
IV	Illumination Variation: the area of the target bounding-box is subject to light variation
SOB	Similar Objects: there are objects in the video of the same object category or with similar appearance to the target
RIG	Rigid Object: the target is a rigid object
DEF	Deformable Object: the target is a deformable object
ROT	Rotation: the target rotates in the video
POC	Partial Occlusion: the target is partially occluded in the video
FOC	Full Occlusion: the target is fully occluded in the video
OUT	Out Of View: the target completely leaves the video frame
MB	Motion Blur: the target region is blurred due to target or camera motion
FM	Fast Motion: the target bounding-box has a motion change larger than its size
LR	Low Resolution: the area of the target bounding-box is less than 1000 pixels in at least one frame
HR	High Resolution: the area of the target bounding-box is larger than 250000 pixels in at least one frame
HM	Head Motion: the person moves their head significantly thus causing camera motion
1H	1 Hand Interaction: the person interacts with the target object with one hand for consecutive video frames
2H	2 Hands Interaction: the person interacts with the target object with both hands for consecutive video frames

is worth noticing that the proposed dataset is competitive in terms of size with respect to the evaluation benchmarks available in the visual (single) object tracking community.

We remark that TREK-150 has been designed for the *evaluation* of visual tracking algorithms in FPV regardless of their methodology. Indeed, in this Chapter, we do not aim to provide a large-scale dataset for the development of deep learning-based trackers. Instead, our goal is to assess the impact of the first-person viewpoint on current trackers. To achieve this goal we follow the standard practice in the visual object tracking community [71, 134, 70, 204, 135, 92, 206] that suggests to set up a small but *well described dataset* to benchmark the tracking progress.

6.2.1 Data Collection

Video Collection. The videos contained in TREK-150 have been sampled from EK [179, 203], which is a public, large-scale, and diverse dataset of egocentric videos focused on human-object interactions in kitchens. This is currently one of the largest datasets for understanding human-object interactions in FPV. Thanks to its dimension, EK provides a significant amount of diverse interaction situations between various people and several different types of objects. Hence, it allows us to select suitable disparate tracking sequences that reflect the common scenarios tackled in FPV tasks. EK offers videos annotated with the actions performed by the camera wearer in the form of temporal bounds and verb-noun labels. The subset of EK known as EK-55 [179] also contains sparse bounding box references of manipulated objects annotated at 2 frames per second in a temporal window around each action. We exploited such a feature to obtain a suitable pool of video sequences interesting for object tracking. Particularly,

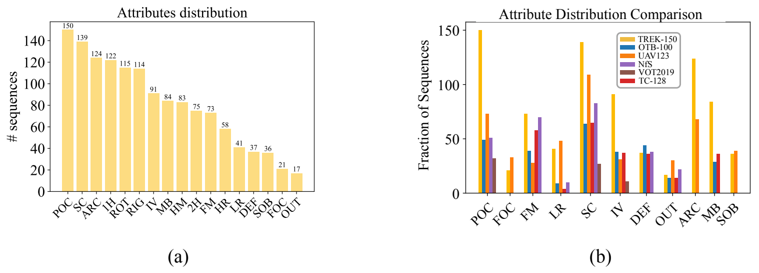


Figure 6.2: (a) Distribution of the sequences within TREK-150 with respect to the attributes used to categorize the visual variability happening on the target object and scene. (b) Comparison of the distributions of common sequence attributes across different benchmarks.

we cross-referenced the original verb-noun temporal annotations of EK-55 to the sparse bounding box labels. This allowed to select sequences in which the camera wearer manipulates an object. Each sequence is composed of the video frames contained within the temporal bounds of the action, extracted at the original 60 FPS frame rate and at the original full HD frame size [179, 203]. From the initial pool, we selected 150 video sequences which were characterized by attributes such as scale changes, partial/full occlusion and fast motion, which are commonly considered in standard tracking benchmarks [71, 70, 37, 38, 92]. The top part of Table 6.2 reports the 13 attributes considered for the selection.

Frame Rate. The videos contained in TREK-150 have a frame rate of 60 FPS and full HD frame size. This is inherited from the EK dataset [179, 203], from which videos are sampled. According to the authors [179, 203], EK has been acquired with such a setting because of the proximity of the camera point of view and the main scene (i.e. manipulated objects), which causes very fast motion and heavy motion blur when the camera wearer moves (especially when he/she moves the head). We empirically evaluated the fast motion issue by assessing the average normalized motion happening on the frames that include fast motion (FM) (we computed them by considering the automatic procedure defined in [71, 37] to assign the FM attribute). Such a motion quantity has been computed as the distance between the center of two consecutive ground-truth bounding-boxes normalized by the frame size. Considering TREK-150 with the videos at 30 FPS, such a value achieves 0.075. This is higher than the 0.068 obtained for OTB-100, the 0.033 of UAV123, or the 0.049 of NfS considered at 30 FPS. These comparisons demonstrate that the FPV scenario effectively includes challenging scenarios due to the faster motion of targets/scene. Considering the 60 FPS frame rate, the fast motion quantity of TREK-150 is reduced to 0.062, which is comparable to the values obtained in other third-person tracking benchmarks.

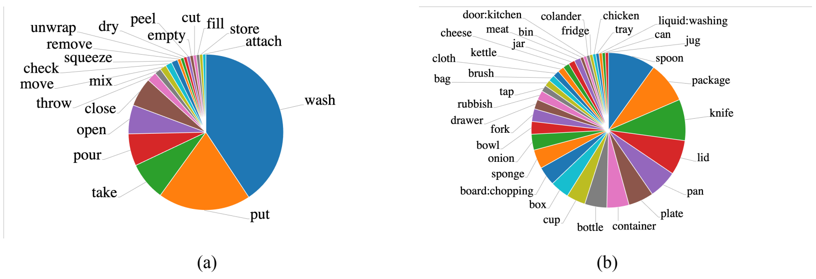


Figure 6.3: Distributions of (a) action verb labels and (b) target object categories.

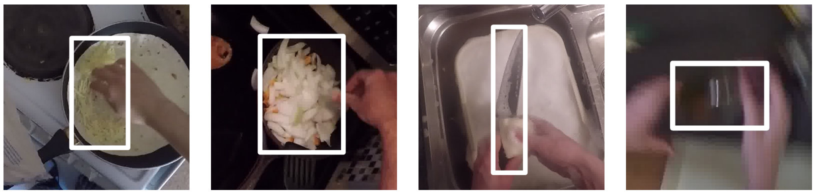


Figure 6.4: Examples of target objects contained in TREK-150 that are difficult to represent with more sophisticated representations (e.g. rotated bounding-box or segmentation mask). The first two images from the left show objects such as "cheese" and "onion" which prevent the determination of the angle for an oriented bounding-box, or an accurate segmentation mask. The last two images present objects which prevent a consistent definition of a segmentation.

6.2.2 Data Labeling

Single Object Tracking. In this study, we restricted our analysis to the tracking of a single target object per video. This has been done because in the FPV scenario a person interacts through his/her hands with one object at a time in general [179, 203]. If a person interacts with two objects at the same time those can be still tracked by two single object trackers. Moreover, focusing on a single object allows us to analyze better all the challenging and relevant factors that characterize the tracking problem in FPV. We believe that future work could investigate the employment of multiple object tracking (MOT) [218, 219] solutions for a general understanding of the position and movement of all objects visible in the scene. We think the in-depth study presented in this Chapter will give useful insights for the development of such methods.

Frame-level Annotations. After selection, the 150 sequences were associated to only 3000 bounding boxes, due to the sparse nature of the object annotations in EK-55. Since it has been shown that visual tracking benchmarks require dense and accurate box annotations [92, 70, 38, 220], we re-annotated the bounding boxes of the target objects on the 150 sequences selected. Batches of sequences were delivered to annotators (21 subjects) who were instructed to perform the labeling. Such initial annotations were then carefully checked and refined by a PhD student, and finally revised by an early-stage researcher and by two professors. This process produced 97296 frames labeled

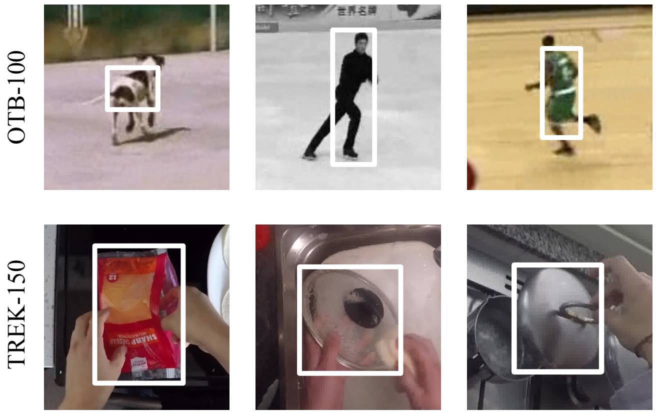


Figure 6.5: Examples of the quality of the bounding-box annotations contained in TREK-150 in comparison with the ones available in the popular OTB-100 benchmark. TREK-150 provides careful and high-quality annotations that tightly enclose all the target objects.

with bounding boxes related to the position and visual presence of objects the camera wearer is interacting with. Following the initial annotations of EK-55, we employed axis-aligned bounding boxes to localize the target objects. This design choice is supported by the fact that such a representation is largely used in many FPV pipelines [221, 222, 182, 179, 223, 192, 224]. Therefore, computing tracking metrics based on such representations allows us to correlate the results with those of object localization pipelines in FPV tasks, ultimately better highlighting the impact of trackers in such contexts. Also, the usage of more sophisticated target representation would have restricted our analysis since the majority of state-of-the-art trackers output just axis-aligned bounding boxes [31, 33, 32, 40, 34, 43, 41, 60, 211, 207, 225, 226, 132, 45, 46, 61, 62, 140, 127, 146, 227, 137, 97, 98, 130, 136, 65, 67, 228], and their recent progress on various benchmarks using such representation [71, 70, 135, 206, 37, 38, 39] proves that it provides sufficient information for tracker initialization and consistent and reliable performance evaluation. Producing high-quality segmentations would have required a great annotation effort, while bounding boxes offer a time-saving yet consistent alternative. Moreover, we point out that many of the objects commonly appearing in FPV scenarios are difficult to annotate consistently with more sophisticated target representations. For these motivations, in this first study we restricted on the analysis of trackers by means of a bounding box representation. We leave for future work the task of assessing the suitability of the segmentation representation for object tracking in FPV. We remark that the proposed bounding boxes have been carefully and tightly drawn around the visible parts of the objects. Figure 6.5 shows some examples of the quality of the bounding-box annotations of TREK-150 in contrast to the ones available in the popular OTB-100 tracking benchmark.

In addition to the bounding boxes for the object to be tracked, TREK-150 provides per-frame annotations of the location of the left and right hand of the camera viewer and of the state of interaction happening between each hand and the target object.

Interaction annotations consist of labels expressing which hand of the camera wearer is currently in contact with the target object (e.g., we used the labels LHI, RHI, BHI to express whether the person is interacting with the target by her/his left or right hand or with both hands). We considered an interaction happening even in the presence of an object acting as a medium between the hand and the target. E.g., we considered the camera wearer to interact with a dish even if a sponge is in between her/his hand and the dish. The fourth row of Figure 6.1 shows a visual example of these situations. These kinds of annotations have been obtained by the manual refinement (performed by the four aforementioned subjects) of the output given by the FPV hand-object interaction detector Hands-in-Contact (HiC) [224].

Sequence-level Annotations. The sequences have been also labeled considering 17 attributes which define the motion and visual appearance changes the target object or the scene is subject to. These are used to analyze the performance of the trackers under different aspects that may influence their execution. The attributes employed in this study include 13 attributes used in standard tracking benchmarks [71, 37, 38], plus 4 additional new ones (High Resolution, Head Motion, 1-Hand Interaction, 2-Hands Interaction) which have been introduced in this Chapter to characterize sequences from FPV-specific point of views. The 17 attributes are summarized in Table 6.2. Figure 6.2(a) reports the distributions of the sequences with respect to the 17 attributes, while Figure 6.2(b) compares the distributions of the most common attributes in the field in TREK-150 and in other well-known tracking benchmarks. Our dataset provides a larger number of sequences affected by partial occlusions (POC), changes in scale (SC) and/or aspect ratio (ARC), motion blur (MB), and fast motion (FM). These peculiarities are due to the particular first person viewpoint and to the human-object interactions which affect the camera motion and the appearance of objects. Based on the verb-noun labels of EK, sequences were also associated to 20 verb labels (e.g., "wash" - see Figure 6.1) and 34 noun labels indicating the category of the target object (e.g., "box"). Figures 6.3(a-b) report the distributions of the videos with respect to verb and target object labels. As can be noted, our benchmark reflects the long-tail distribution of labels in EK [179].

6.2.3 Differences With Other Tracking Benchmarks

We believe that the proposed TREK-150 benchmark dataset offers complementary features with respect to existing visual tracking benchmarks.

Table 6.1 and Figure 6.2(a) and (b) show that TREK-150 provides complementary characteristics to what is available today to study the performance of visual trackers. Particularly, our proposed dataset offers different distributions of the common challenging factors encountered in other datasets. For example, TREK-150 includes a larger number of examples with occlusions (POC), fast motion (FM), scale change (SC), aspect ratio change (ARC), illumination variation (IV), and motion blur (MB), while it provides a competitive number of scenarios for low resolution (LR), full occlusion (FOC), deformable objects (DEF), and presence of similar objects (SOB). Additionally, even though the 4 new attributes high resolution (HR), head motion (HM), one-hand interaction (1H), two-hands interaction (2H), define particular FPV scenarios, we think that

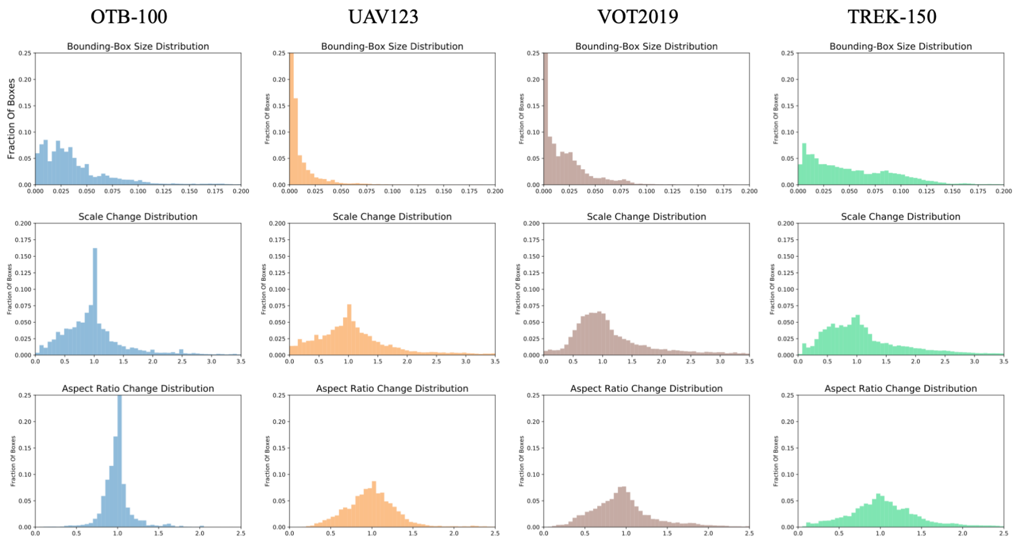


Figure 6.6: Comparison between TREK-150 (last column of plots) and other popular visual tracking benchmarks on the distributions computed for different bounding-box characteristics. Each column of plots reports the distribution of bounding-box sizes, scale-changes, and aspect ratio change (the x-axis of each plot reports the range of the bounding-box statistic).

they can be of interest even for the visual tracking community. For example, as shown by the second row of images of Figure 6.5, 1H and 2H can be considered as attributes that define different levels of occlusion, as objects manipulated with two hands generally cause more extended hiding of the targets. Besides these sequence-level features, TREK-150 offers up to 34 target categories which, to the best of our knowledge, have never been studied. As shown by the Figures 6.4 and 6.5, these objects have challenging appearances (e.g. transparent or reflective objects like lids, bottles, or food boxes) and shapes (e.g. knives, spoons, cut food) that change dramatically due to the interaction or motion induced by the camera viewer.

We additionally computed some statistics on the bounding-box ground-truth trajectories contained in the proposed dataset. Figure 6.6 reports these distributions. For comparison, we report the distributions computed on the popular tracking benchmarks VOT2019, UAV123, OTB-100. As can be noted, our dataset exhibits different distributions, and thus offers different behaviors of the target appearances and motions. Particularly, observing the first plot of the last column, it can be noted that TREK-150 has a wider distribution of bounding-box sizes, hence making it suitable for the evaluation of trackers with targets of many different sizes. Particularly, TREK-150 has a larger number of bounding-boxes with greater dimension. The plot just below the first shows that TREK-150 provides more references to assess the trackers' capabilities in tracking objects that become smaller. Finally, the last plot shows a wider distribution for the aspect ratio change, showing that TREK-150 offers a large variety of examples to evaluate the capabilities of trackers in predicting the shape change of targets.

Additionally to these characteristics, we think TREK-150 is interesting because it allows the study of visual object tracking in unconstrained scenarios of every-day situations.

Table 6.3: Characteristics of the generic object trackers considered in our evaluation. We provide details about: the Image Representation employed by the trackers (Pixel column - ✓if the tracker uses raw pixel intensity values; HOG column - ✓if the tracker uses Histogram of Oriented Gradients; Color column - ✓if the tracker uses Color Names or Intensity; CNN column - the Convolutional Neural Network backbone used); the Matching operation performed to find the target in sequence frames (CF column - ✓if the tracker uses correlation filters; CC column - ✓if the tracker uses the cross correlation; Concat column - ✓if the tracker concatenates features; T-by-D column - ✓if the tracker uses a tracking-by-detection approach; Had column - ✓if the tracker uses hadamard correlation; Tra column - ✓if the tracker uses a transformer-based correlation). The ✓ symbol in the Model Update column expresses whether the tracker updates the target model during the tracking procedure. The last four columns report the category of tracking approach according to [229] (ST_0 column - short-term trackers without any re-detection mechanism; ST_1 column - short-term trackers without any re-detection mechanism but that estimate tracking confidence; LT_0 column - pseudo long-term trackers that do not detect failure and do not perform explicit re-detection; LT_1 column - long-term trackers that detect tracking failure and perform re-detection).

Tracker	Venue	Image Representation				Matching Operation						Model	Class given by [229]			
		Pixel	HOG	Color	CNN	CF	CC	Concat	T-by-D	Had	Tra	Update	ST_0	ST_1	LT_0	LT_1
MOSSE [31]	CVPR 2010	✓				✓						✓	✓			
DSST [33]	BMVC 2014	✓	✓			✓						✓	✓			
KCF [32]	TPAMI 2015		✓			✓						✓	✓			
MDNet [40]	CVPR 2016				VGG-M				✓			✓		✓		
Staple [34]	CVPR 2016		✓	✓		✓						✓	✓			
SiamFC [43]	ECCVW 2016				AlexNet		✓						✓			
GOTURN [41]	ECCV 2016				AlexNet			✓					✓			
ECO [60]	CVPR 2017				VGG-M	✓						✓		✓		
BACF [211]	ICCV 2017		✓			✓						✓	✓			
DCFNet [230]	ArXiv 2017				VGG-M	✓						✓	✓			
VITAL [207]	CVPR 2018				VGG-M				✓			✓		✓		
STRCF [225]	CVPR 2018		✓			✓						✓	✓			
MCCTH [226]	CVPR 2018		✓	✓		✓						✓	✓			
DSLT [231]	ECCV 2018				VGG-16		✓					✓	✓			
MetaCrest [132]	ECCV 2018				VGG-M	✓						✓		✓		
SiamRPN++ [45]	CVPR 2019				ResNet-50		✓						✓			
SiamMask [47]	CVPR 2019				ResNet-50		✓						✓			
SiamDW [46]	CVPR 2019				ResNet-22		✓						✓			
ATOM [61]	CVPR 2019				ResNet-18	✓						✓		✓		
DiMP [62]	ICCV 2019				ResNet-50	✓						✓		✓		
SPLT [140]	ICCV 2019				ResNet-50	✓						✓				✓
UpdateNet [232]	ICCV 2019				AlexNet		✓					✓	✓			
SiamFC++ [227]	AAAI 2020				AlexNet		✓						✓			
GlobalTrack [146]	AAAI 2020				ResNet-50					✓					✓	
PrDiMP [127]	CVPR 2020				ResNet-50	✓						✓		✓		
SiamBAN [97]	CVPR 2020				ResNet-50		✓						✓			
D3S [124]	CVPR 2020				ResNet-50	✓							✓			
LTMU [137]	CVPR 2020				ResNet-50	✓	✓		✓			✓				✓
Ocean [98]	ECCV 2020				ResNet-50		✓						✓			
KYS [130]	ECCV 2020				ResNet-50	✓						✓		✓		
TRASFUST [142]	ACCV 2020				ResNet-18			✓					✓			
SiamGAT [136]	CVPR 2021				ResNet-50		✓						✓			
TrDiMP [?]	CVPR 2021				ResNet-50	✓					✓	✓		✓		
LightTrack [233]	CVPR 2021				NAS		✓						✓			
TransT [66]	CVPR 2021				ResNet-50						✓	✓				
STMTrack [228]	CVPR 2021				GoogLeNet			✓				✓	✓			
STARK [67]	ICCV 2021				ResNet-50						✓	✓		✓	✓	
KeepTrack [151]	ICCV 2021				ResNet-50	✓						✓		✓	✓	

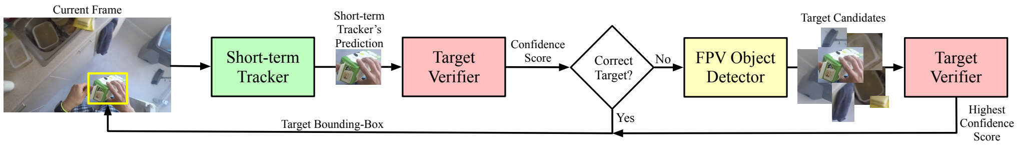


Figure 6.7: Scheme of execution of the proposed FPV baseline trackers LTMU-F and LTMU-H based on LTMU [137].

6.3 Trackers

6.3.1 Generic Object Trackers

Among the examined trackers, 38 have been selected to represent different popular approaches to generic-object visual tracking. Specifically, in the analysis we have included short-term trackers [229] based on both correlation-filters with hand-crafted features (MOSSE [31], DSST [33], KCF [32], Staple [34], BACF [211], DCFNet [230], STRCF [225], MCCTH [226]) and deep features (ECO [60], ATOM [61], DiMP [62], PrDiMP [127], KYS [130], KeepTrack [151]). We also considered deep siamese networks (SiamFC [43], GOTURN [41], DSLT [231], SiamRPN++ [45], SiamDW [46], UpdateNet [232], SiamFC++ [227], SiamBAN [97], Ocean [98], SiamGAT [136], STMTrack [228]), tracking-by-detection methods (MDNet [40], VITAL [207]), as well as trackers based on target segmentation representations (SiamMask [47], D3S [124]), meta-learning (MetaCrest [132]), fusion of trackers (TRASFUST [142]), neural architecture search (LightTrack [233]), and transformers (TrDiMP [65], TransT [66], STARK [67]). The long-term [229] trackers SPLT [140], GlobalTrack [146], and LTMU [137] have been also taken into account in the study. These kinds of trackers are designed to address longer target occlusion and out of view periods by exploiting an object re-detection module. All of the selected trackers are state-of-the-art approaches published between the years 2010-2021. Table 6.3 reports detailed information about the 38 considered generic-object trackers regarding the: venue and year of publication; type of image representation used; type of target matching strategy; employment of target model updates; and category of tracker according to the classification of [229]. For each tracker, we used the code publicly available and adopted default parameters for evaluation purposes. The code was run on a machine with an Intel Xeon E5-2690 v4 @ 2.60GHz CPU, 320 GB of RAM, and an NVIDIA TITAN V GPU.

6.3.2 FPV Trackers

Since there are no public implementations of the FPV trackers described in Section 6.1.1 and given also that they were not designed to track generic objects, we developed 2 FPV-specific generic object trackers in addition to the aforementioned ones. Particularly, they combine the LTMU tracker [137] with FPV-specific object detectors. The first solution, referred to as LTMU-F, employs the Faster-R-CNN object detector trained on EK-55 and provided by the authors of [179], while the second, denoted as LTMU-H, uses the Faster-R-CNN-based hand-object detector HiC [224]. These baseline trackers exploit the respective detectors as re-detection modules according to the LTMU scheme [137].

For a better understanding, we briefly recap the processing procedure of the LTMU tracker [137]. After being initialized with the target in the first frame of a sequence, at every other frame LTMU first executes the short-term tracker DiMP [62] that tracks the target in a local area of the frame based on the target's last position. The patch extracted from the box prediction of DiMP is evaluated by an online-learned verification module based on MDNet [40], which outputs a probability estimate of the target being contained in the patch. Such an estimate is used to decide if the short-term tracker is tracking the target or not. If it is, its predicted box is given as output for the current frame. In the other case, a re-detection module is executed to look for the target in the whole frame. The re-detector returns some candidate locations which may contain the target and each of these is checked by the verification module. The candidate patch with the highest confidence is given as output and used as a new target location to re-initialize DiMP. In our settings, we employed FPV-based detectors to implement such a re-detection module. For LTMU-F, such a module has been set to retain the first 10 among the many detections given as output, considering a ranking based on the scores attributed by the detector to each detection. If no detection is given for a frame, the last available position of the target is considered as a candidate location. For LTMU-H, we used the object localizations of the hand-object interaction detections given by the FPV version of HiC [224] as target candidate locations. HiC is implemented as an improved Faster R-CNN which is set to provide, at the same time, the localization of hands and interacted objects, as well as their state of interaction. As for LTMU-F, if no detection is given for a frame, the last available position of the target is considered as a candidate location. For both detection methods, the original pre-trained models provided by the authors have been used. The described setups, the common scheme of which is presented in Figure 6.7, give birth to two new FPV trackers that implement conceptually different strategies for FPV-based object localization and tracking. Indeed, the first solution aims to just look for objects in the scene, while the second one reasons in terms of the interaction happening between the camera wearer and the objects.

The choice of using LTMU [137] as a baseline methodology stems from its highly modular scheme which makes it the most easily configurable tracker with state-of-the-art performance available today. We took advantage of the commodity of a such framework to insert the FPV-specific modules described before.

6.4 Evaluation Settings

6.4.1 Evaluation Protocols

The protocols used to execute the trackers are described in the following.

One-Pass Evaluation. We employed the one-pass evaluation (OPE) protocol detailed in [71] which implements the most realistic way to run a tracker in practice. The protocol consists of two main stages: (i) initializing a tracker with the ground-truth bounding box of the target in the first frame; (ii) letting the tracker run on every subsequent frame until the end of the sequence and record predictions to be considered for the evaluation. To obtain performance scores for each sequence, predictions and ground-truth bounding boxes are compared according to some distance measure only

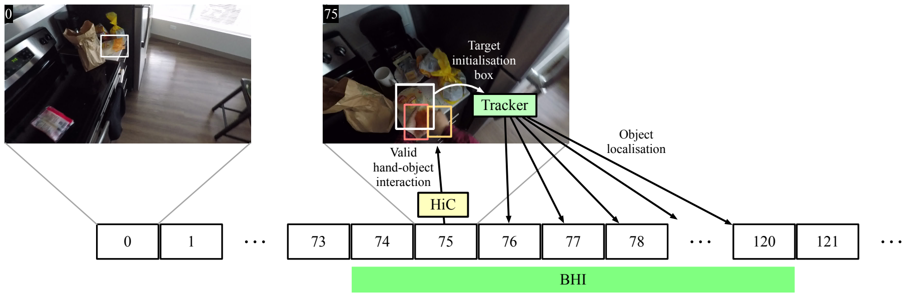


Figure 6.8: Schematic visualization of the protocol designed to execute trackers in the context of a hand-object interaction detection task. The hand-object interaction labels provided for TREK-150 are used to consider sub-sequences of frames in which the camera wearer is interacting with the target object. In this picture, the labels BHI are employed to indicate that an interaction by both hands is happening in the frame range [74, 120]. On such sub-sequences, a systematic pipeline for hand-object interaction detection and tracking is run. The hand-object interaction detector HiC [224] is first executed in every frame to obtain a valid hand-object interaction (in this example the first valid detection is obtained at frame 75). Once such an event is determined, the tracker is initialized with the bounding box given by HiC for the object involved in the interaction. The tracker is then run on all the subsequent frames to provide the reference to such an object.

in frames where ground-truths are present (ground-truth bounding boxes are not given for frames in which the target is fully occluded or out of the field of view). The overall scores are obtained by averaging the scores achieved for every sequence.

Multi-Start Evaluation. To obtain a more robust evaluation [94], especially for the analysis over sequence attributes and action verbs, we employed the recent protocol proposed in [119], which defines different points of initialization along a video. In more detail, for each sequence, different initialization points –called anchors– separated by 2 seconds are defined. Anchors are always set in the first and last frames of a sequence. Some of the inner anchors are shifted forward by a few frames in order to avoid frames in which the target is not visible. A tracker is run on each of the sub-sequences yielded by the anchor either forward or backward in time depending on the longest sub-sequence the anchor generates. The tracker is initialized with the ground-truth annotation in the first frame of the sub-sequence and let run until its end. Then, as for the OPE, predicted and ground-truth boxes are compared to obtain performance scores for each sub-sequence. Scores for a single sequence are computed by averaging the scores of each sub-sequence weighted by their length in number of frames. Similarly, the overall scores for the whole dataset are obtained by averaging each sequence's score weighted by its number of frames. We refer to this protocol as multi-start evaluation (MSE). It allows a tracker to better cover all the situations happening in the sequences, ultimately leading to more robust evaluation scores.

Real-Time Evaluation. Since many FPV tasks such as object interaction [183] and early action recognition [222], or action anticipation [179], require real-time computation, we evaluate trackers in such a setting by following the details given in [84, 234]. This protocol, which we refer to as RTE, is similar to OPE. A tracker is initialized with the ground-truth in the first frame of a sequence. Then the algorithm is presented with a new frame only after its execution over the previous frame has finished. The new presented frame is the last frame available for the time instant in which the tracker becomes ready to be executed, considering that frames occur regularly based on the frame rate of the video. In other words, all the frames occurring in the time interval between the start and end time instants of the tracker's execution are skipped. For all such frames, the last box given by the tracker is used as location for the target. The overall performance scores are ultimately obtained as for the OPE protocol.

FPV Downstream Task Evaluation. We also evaluated trackers in the context of a video-based hand-object interaction detection solution in order to assess their direct impact on a downstream FPV-specific task. The aim of this task is to determine when and where in the frames the camera wearer is interacting (e.g., by touching/manipulating) with an object with his/her hands. To achieve the goal, we built a solution composed of a HiC instance [224] to detect the hands and their state of interaction with an object and a visual tracker to maintain the reference to it. To evaluate their impact on this downstream task, we execute the trackers in two ways: (i) we initialize the tracker using the object bounding box prediction given by HiC [224] in the first valid hand-object interaction detection; (ii) we initialize the tracker using the ground-truth bounding box for the target in the first frame in which a hand-object interaction label is present (this setup is equivalent to having an optimal hand-object interaction detector). A graphical representation of the execution of the described pipeline is given in Figure 6.8. Taking inspiration from the metric used by [224] to evaluate the performance of HiC on static images, we quantify the performance of the proposed pipeline by the normalized count of frames in which the given hand-object detection matches the ground-truth annotation available. Such matching is said to happen when the bounding box predictions for the hands have an intersection-over-union (IoU) ≥ 0.5 with the hand ground-truth boxes, the predicted interaction state is "in contact", and the object bounding box has an IoU ≥ 0.5 with the ground-truth box [224]. For our experiments, we restricted the analysis of the solution on the sub-sequences contained in TREK-150 in which a hand-object interaction is present. These are determined by considering the sub-sequences of consecutive frames having the same interaction label (i.e., LHI, RHI, BHI). To obtain an overall performance score, which we refer to as Recall, we average the sub-sequence scores after having them weighted by the sub-sequence lengths in number of frames, in a similar fashion as we did to compute score in the MSE.

This experimental procedure gives us an estimate of the accuracy of the hand-object interaction detection system under configurations with different trackers. More interestingly, the proposed evaluation protocol allows also to build a ranking of the trackers based on the results of a downstream application. To the best of our knowledge, this setup brings a new way to assess the performance of visual object trackers.

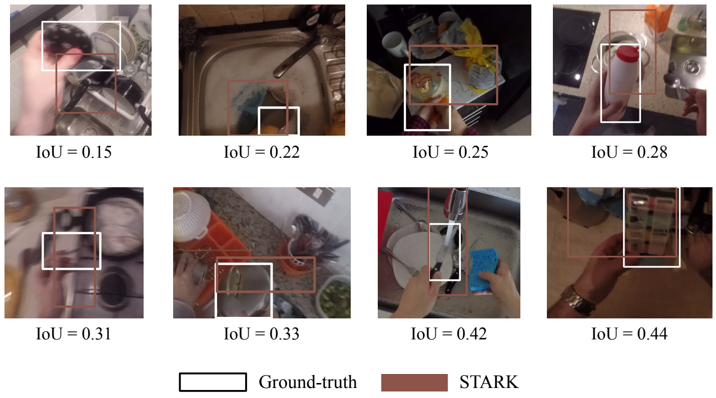


Figure 6.9: Visual examples of very imprecise tracker predictions (given by STARK). Such bounding boxes do not locate well the ground-truth target object and can be considered as wrong target positions. But considering that the IoUs are greater than the 0.1 threshold, such outputs result as positive predictions in the standard Robustness measure [119]. By considering multiple thresholds, the proposed GSR score allows to obtain a single metric capturing all of the visualized situations as wrong predictions.

6.4.2 Performance Measures

To quantify the performance of the trackers, we used different measures that compare trackers' predicted bounding boxes with the temporally aligned ground-truth boxes. To evaluate the overall localization accuracy of the trackers, we employ the success plot [71], which shows the percentage of predicted boxes whose IoU with the ground-truth is larger than a threshold varied from 0 to 1 (Figure 6.10 (a)). We also use the normalized precision plot [37], that reports, for a variety of thresholds, the percentage of boxes whose center points are within a given normalized distance from the ground-truth (Figure 6.10 (b)). As summary measures, we report the success score (SS) [71] and normalized precision scores (NPS) [37], which are computed as the Area Under the Curve (AUC) of the success plot and normalized precision plot respectively.

Along with these standard metrics, we employ a novel plot which we refer to as generalized success robustness plot (Figure 6.10 (c)). For this, we take inspiration from the robustness metric proposed in [119] which measures the normalized extent of a tracking sequence before a failure. We believe this aspect to be especially important in FPV as a superior ability of a tracker to maintain longer references to targets can lead to the better modeling of actions and interactions. The original metric proposed in [119] uses a fixed threshold of 0.1 on the bounding box overlap to detect a collapse of the tracker. Such a value was determined mainly to reduce the chance of cheating in the VOT2020 competition and it is not necessarily the case that such a value could work well for different tracking applications. Indeed, as can be visualized in Figure 6.9, even IoUs greater than 0.1 can be associated to bounding box predictions wrongly located with respect to the position and scale of the target objects. Therefore, such a fixed threshold limits the detection of the failure cases of a tracker. To generalize the Robustness metric [119], we take inspiration from the success and normalized precision plots and propose to use different box overlap thresholds ranging in [0, 0.5] to determine the collapse. We consider 0.5 as the maximum threshold as higher overlaps are usually associated to positive predictions in many computer vision tasks. Overall, our proposed plot allows to assess the length of tracking sequences in a more general way that is better aligned with the requirements of different application scenarios including FPV ones. Similarly to [71, 37], we use the AUC of the generalized success robustness plot to obtain an aggregate score which we refer to as generalized success robustness (GSR).

Together with the SS, NPS, and GSR results achieved with RTE protocol results, we evaluate the trackers' processing speed in frames per second (FPS) to quantify their efficiency.

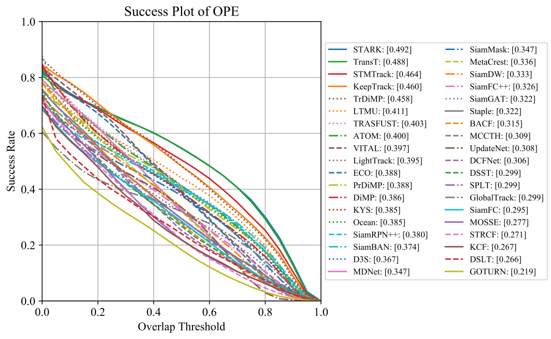
6.5 Results

6.5.1 Performance of Generic Object Trackers

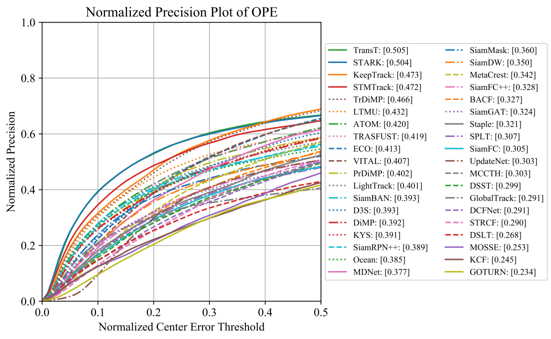
Figure 6.10 reports the performance of the 38 generic object trackers on TREK-150 using the OPE protocol, while Figure 6.11 present the results achieved with the MSE protocol. Figure 6.12 presents examples that qualitatively show the performance of some of the trackers. Considering the results on a tracking approach basis, we have that trackers based on deep learning (e.g. STARK, TransT, KeepTrack, LTMU, TrDiMP, ATOM, VITAL, ECO, Ocean) perform better than those based on hand-crafted features (e.g. BACF, MCCTH, DSST, KCF). Among the first class of trackers, the ones leveraging online adaptation mechanisms (e.g. STARK, STMTrack, KeepTrack, LTMU, TrDiMP, ATOM, VITAL, ECO, KYS, DiMP) are more accurate than the ones based on single-shot instances (e.g. SiamGAT, Ocean, D3S, SiamBAN, SiamRPN++). Trackers based on the transformer architecture [64] (e.g. STARK, TransT, TrDiMP) hold the highest positions in the rankings of all the plots, suggesting that the representation learning and matching approach exploited by such trackers is suitable for better target-background discrimination in the FPV setting. Indeed, the transformer-based matching operation between template and searching areas like the one implemented by STARK and TransT leads to a higher bounding box overlap on average (SS performance of Figure 6.10(a)) and to a better centered bounding box (NPS performance of Figure 6.10(b)).

Generally, the generalized success robustness plot in Figure 6.10(c) and the GSR results of Figure 6.11 report different rankings of the trackers, showing that more spatially accurate trackers are not always able to maintain their accuracy for longer periods of time. Trackers that aim to build robust target models via online methods (e.g. STMTrack, ECO, TrDiMP, VITAL, MDNet, ATOM) result in better solutions for keeping longer temporal reference to objects. Particularly, the results achieved by STMTrack tell that a strategy based on memory networks building a highly dynamic representation of the template during tracking is beneficial to maintain a longer reference to the target.

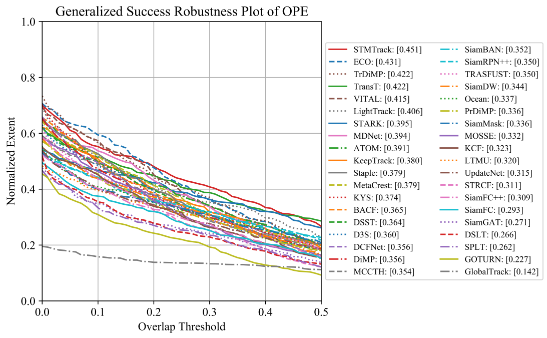
By comparing the performance of the selected trackers with the results they achieve on standard benchmarks such as OTB-100 [71], as reported in Figure 6.13, it can be noticed that the overall performance of all the trackers is decreased across all measures when considering the FPV scenario. These outcomes demonstrate that the FPV setting poses new challenges to the visual trackers currently available.



(a)



(b)



(c)

Figure 6.10: Performance of the 38 selected generic object trackers on the proposed TREK-150 benchmark under the OPE protocol. In brackets, next to the trackers' names, we report the SS, NPS, and GSR values.

6.5.2 Processing Speed Study

Table 6.4 reports the FPS performance of the trackers and the SS, NPS, and GSR scores achieved under the RTE protocol. None of the trackers achieve the frame rate

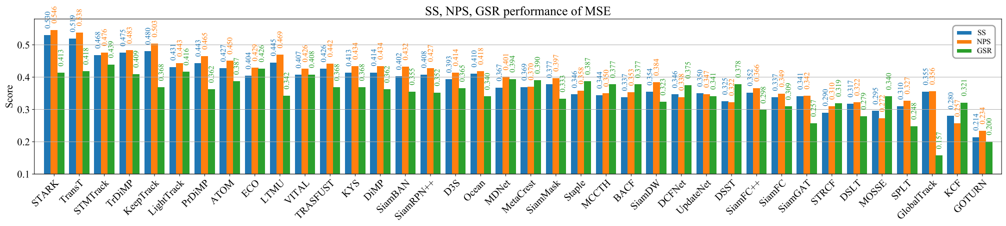


Figure 6.11: SS, NPS, and GSR performance of the 38 benchmarked generic object trackers on the proposed TREK-150 benchmark achieved under the MSE protocol. The trackers are ordered by the average value of their SS, NPS, GSR scores.

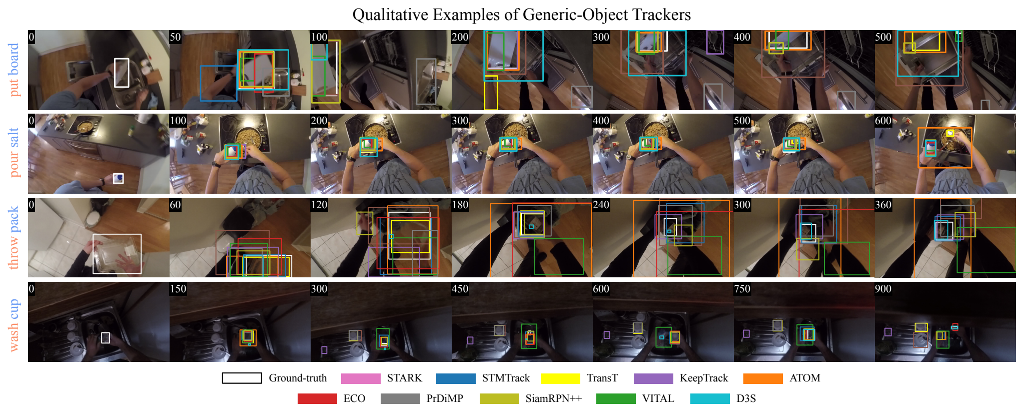


Figure 6.12: Qualitative results of some of the generic object trackers benchmarked on the proposed TREK-150 dataset.

speed of 60 FPS. We argue that this is due to the full HD resolution of frames which requires demanding image crop and resize operations with targets of considerable size. Taking into consideration the tracking approaches, we observe that trackers based on single-shot siamese networks (e.g. Ocean, SiamBAN, SiamRPN++) or on light online adaptation techniques such as the target template change (e.g. as performed by TransT, STARK, STMTrack) emerge as the fastest trackers and exhibit a less significant performance drop of the proposed scores. In particular, the decrease in SS of SiamBAN and SiamRPN++ is of 3.7% and 4.7%, respectively, with respect to the OPE results. TransT and STARK exhibit a larger performance drop of 5.3 % and 6.1% respectively, but their robustness makes their overall real-time performance much higher than SiamBAN and SiamRPN++. Due to the reliance on heavier online learning mechanisms, trackers like KeepTrack, PrDiMP, ATOM, KYS, ECO achieve a lower processing speed that consequently causes a major accuracy loss in real-time scenarios.

In general, we observe that the GSR score is the measure on which all trackers present the major performance drop in the real-time setting, suggesting that particular effort should be spent to make trackers better address longer references to objects in real-time scenarios. Overall, we can say that trackers like TransT and STARK are currently the

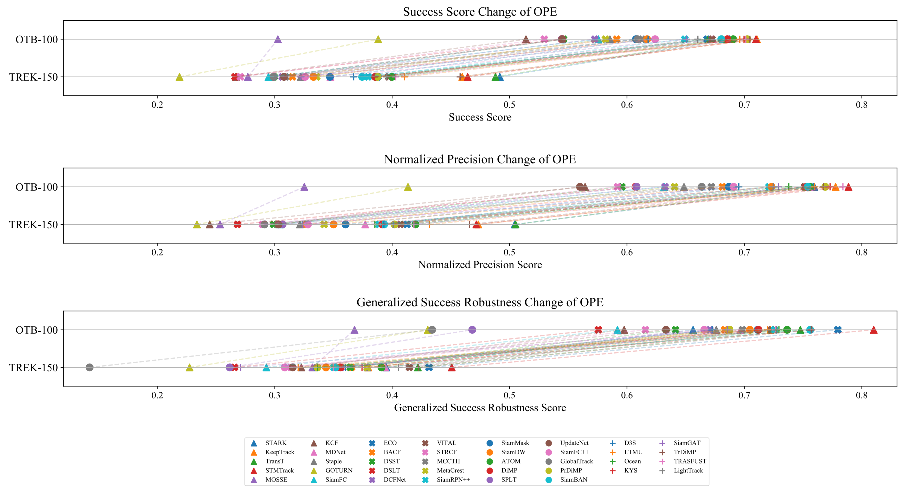


Figure 6.13: Performance comparison of SS, NPS, and GSR scores obtained by the 38 benchmarked generic object trackers on the popular OTB-100 benchmark [71] and on the proposed TREK-150 benchmark under the OPE protocol.

most suitable methods to employ for the development of real-time FPV applications requiring object tracking. Given their limited performance decrease between the OPE and RTE results, siamese-based trackers could serve as promising alternatives if their tracking accuracy and robustness are improved.

6.5.3 Performance of Fine-Tuned Trackers

Trackers such those based on siamese neural networks (e.g. SiamFC, SiamRPN++, SiamBAN, SiamGAT) are said to be offline because they are trained to track objects on large-scale tracking datasets [19, 37, 39, 38] and do not use online adaptation mechanisms at test time. In our evaluation, such trackers have been employed as trained as it is described in their original paper. But given their reduced performance, we performed experiments to understand if their behavior can be improved by learning tracking in the FPV domain. However, our TREK-150 dataset is designed to evaluate the progress of visual tracking solutions in FPV and does not provide a large-scale database of learning examples as needed by these methods (a large-scale dataset for the training of FPV-specific trackers is out of the scope of this study). In this view, TREK-150 well aligns with real-world datasets where millions of frames are not available for training. In such scenarios, the reasonable options the machine learning community suggests are to use the deep learning models as they are because of their general knowledge, or to adapt them through fine-tuning or domain adaptation strategies using a smaller training set. We tried all these options. The results for the first one were given in Section 6.5.1. We carried out the second strategy by randomly splitting TREK-150 into a training and a test set of 100 and 50 videos respectively. We fine-tuned the popular offline trackers SiamFC and SiamRPN++ on such training set according to their original learning

Table 6.4: Performance achieved by the 38 generic object trackers benchmarked on TREK-150 using the RTE protocol. Best results, per measure, are highlighted in **red**, second-best in **blue**, third-best in **green**.

Tracker	FPS	SS	NPS	GSR
TransT	19	**0.462**	**0.471**	**0.394**
STARK	14	**0.453**	**0.456**	**0.345**
STMTrack	13	**0.434**	**0.440**	**0.407**
TrDiMP	9	0.389	0.378	0.287
LightTrack	8	0.376	0.373	0.335
Ocean	21	0.365	0.358	0.294
SiamRPN++	23	0.362	0.356	0.293
SiamBAN	24	0.360	0.369	0.313
PrDiMP	13	0.352	0.349	0.243
KeepTrack	9	0.345	0.335	0.188
DiMP	16	0.336	0.331	0.224
SiamMask	23	0.335	0.333	0.298
SiamFC++	**45**	0.330	0.331	0.308
SiamDW	32	0.327	0.334	0.317
KYS	12	0.327	0.317	0.219
ATOM	15	0.319	0.312	0.179
SiamGAT	20	0.314	0.306	0.257
UpdateNet	21	0.311	0.297	0.295
DCFNet	**49**	0.299	0.286	0.335
TRASFUST	13	0.296	0.270	0.185
SiamFC	34	0.293	0.295	0.280
D3S	16	0.276	0.263	0.182
BACF	9	0.276	0.262	0.234
SPLT	8	0.265	0.247	0.203
STRCF	10	0.264	0.250	0.218
DSLT	7	0.260	0.234	0.211
ECO	15	0.252	0.231	0.173
GlobalTrack	8	0.253	0.227	0.139
MCCTH	8	0.251	0.231	0.232
Staple	13	0.249	0.236	0.234
GOTURN	**44**	0.247	0.242	0.119
MOSSE	26	0.227	0.190	0.244
LTMU	3	0.213	0.178	0.161
MetaCrest	8	0.207	0.175	0.165
VITAL	4	0.204	0.165	0.158
DSST	2	0.191	0.145	0.161
KCF	6	0.186	0.157	0.177
MDNet	1	0.185	0.140	0.161

strategy. We then tested the fine-tuned versions on the test set and the results are reported in Table 6.5 in comparison with the original counterparts. The results show that the simple fine-tuning leads to substantial overfitting that cause the performance to drop in general. The exceptions are given by SiamRPN++'s NPS results which are increased through such adaptation procedure. Table 6.6 report the results of the weakly-supervised domain adaptation methodology based on knowledge distillation and reinforcement learning proposed in Chapter 5. As for the fine-tuning experiment, we considered 100 videos for training and 50 for testing. The STARK, KeepTrack, and VITAL trackers have been employed as teacher trackers. The results demonstrate that the proposed domain adaptation solution works well for the improvement of the baseline performance of the TRAS tracker (the latter presented in Chapter 3). According to the OPE protocol, using the proposed adaptation method (TRAS-DA) results better than employing a fine-tuning approach (TRAS-FT), while according to the MSE protocol the two adaptation strategies result comparable. The outcomes of this experiment are the following. Particular adaptation strategies could have potential if implemented over

Table 6.5: Performance of the offline trackers SiamFC and SiamRPN++ on a subset of 50 sequences of TREK-150 without and with fine-tuning on the remaining 100 videos.

Tracker	Fine-tuning	OPE			MSE		
		SS	NPS	GR	SS	NPS	GR
SiamFC	✗	0.311	0.332	0.317	0.307	0.317	0.307
	✓	0.267	0.275	0.278	0.287	0.305	0.292
SiamRPN++	✗	0.384	0.395	0.377	0.367	0.385	0.333
	✓	0.348	0.407	0.313	0.336	0.406	0.314

Table 6.6: Performance of the deep regression tracker TRAS (presented in Chapter 3) adapted for the FPV domain by fine-tuning with a standard fully supervised regression loss [41] (TRAS-FT) and by the weakly-supervised domain adaptation strategy based on knowledge distillation and reinforcement learning presented in Chapter 5 (TRAS-DA). Both strategies were tested using a subset of 50 sequences of TREK-150 and the remaining 100 videos for training.

Tracker	OPE			MSE		
	SS	NPS	GSR	SS	NPS	GSR
TRAS	0.201	0.188	0.196	0.206	0.187	0.159
TRAS-FT	0.232	0.241	0.258	0.248	0.267	0.258
TRAS-DA	0.254	0.285	0.253	0.246	0.293	0.219

particular neural network architectures for tracking. Given their limited performance even after adaptation, deep regression trackers such as TRAS do not offer a strong baseline in this direction.

6.5.4 Performance of the FPV-specific Trackers

The performances of the proposed FPV baseline trackers LTMU-H and LTMU-F are reported in Table 6.7 in comparison with the original LTMU. Figure 6.14 presents some qualitative examples of the performance of the two FPV-specific trackers in contrast to the original one. Under both the OPE and MSE protocols, the LTMU-H and LTMU-F trackers are largely better than the baseline LTMU in SS, NPS, and GSR. Particularly, the improvement of LTMU-H over LTMU is greater than the 10% across all the performance measures. Considering the RTE protocol, the results achieved by the two new trackers are comparable to the baseline.

Compared to the best trackers discussed in Section 6.5.1, LTMU-H and LTMU-F demonstrate a lower performance. This can be attributed to the tracking ability of the underlying DiMP instance which, as shown in Figure 6.10, is more affected by the challenges introduced by FPV. Nevertheless, the message to take from these outcomes is that adapting a state-of-the-art method with FPV-specific components allows to increase the tracking performance. We hence expect significant performance improvements to be achievable by a tracker accurately designed to exploit FPV-specific cues such as the characteristics of the interaction between the target and the camera wearer.

6.5.5 Attribute Analysis

Figure 6.15 reports the SS, NPS, and GSR scores, computed with the MSE protocol, of the whole selection of 40 trackers with respect to the attributes introduced in Table

Table 6.7: Performance of the proposed baseline FPV-trackers LTMU-H and LTMU-F in comparison with LTMU under the different protocols used for the evaluation on TREK-150. The percentage values in brackets report the performance gain/decrease of each baseline with respect to LTMU.

Protocol		OPE	
Tracker	SS	NPS	GSR
LTMU	0.411	0.432	0.320
LTMU-F	0.456 (+10.9%)	0.477 (+10.4%)	0.372 (+16.3%)
LTMU-H	0.461 (+12.2%)	0.486 (+12.5%)	0.376 (+17.5%)
Protocol		MSE	
Tracker	SS	NPS	GSR
LTMU	0.445	0.469	0.342
LTMU-F	0.485 (+9.0%)	0.508 (+8.3%)	0.375 (+9.7%)
LTMU-H	0.495 (+11.2%)	0.517 (+10.2%)	0.380 (+11.1%)
Protocol		RTE	
Tracker	SS	NPS	GSR
LTMU	0.213	0.178	0.161
LTMU-F	0.205 (-3.8%)	0.161 (-9.6%)	0.162 (+0.6%)
LTMU-H	0.213 (+0%)	0.174 (-2.2%)	0.161 (+0%)

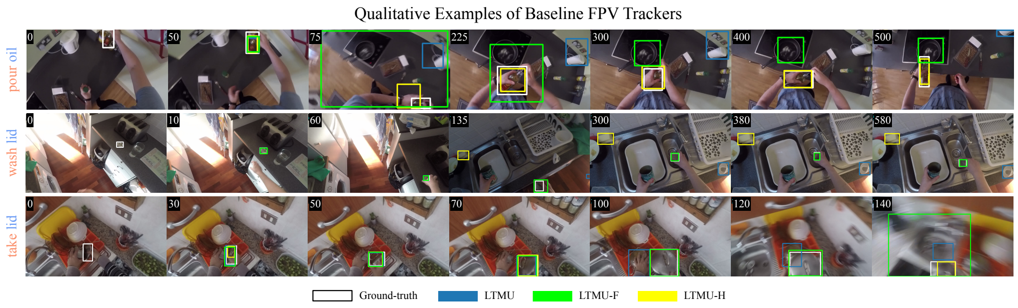


Figure 6.14: Qualitative results of the baseline FPV trackers LTMU-F and LTMU-H in comparison with LTMU.

6.2. We do not report results for the POC attribute as it is present in every sequence, as shown in Figure 6.2 (a). It stands out clearly that full occlusion (FOC), out of view (OUT) and the small size of targets (LR) are the most difficult situations for all the trackers. The fast motion of targets (FM) and the presence of similar objects (SOB) are also critical factors that cause drops in performance. Rotations (ROT) and the illumination variation (IV) are better addressed by the trackers. The algorithms also do not demonstrate significant behavior changes between the tracking of rigid or deformable objects. With respect to the new 4 sequence attributes related to FPV, the results report that tracking objects held with two hands (2H) is more difficult than tracking objects held with a single hand (1H). This is because the manipulation of the target by two hands generates situations in which the occlusions are more extended over the object's appearance. Trackers are instead quite robust to the head motion (HM), which influences the camera movements, and seem to cope well with objects appearing in larger sizes (HR).

In terms of algorithmic principles, we have that STARK has better SS results over the second-best, TransT, across all the conditions described by the attributes except

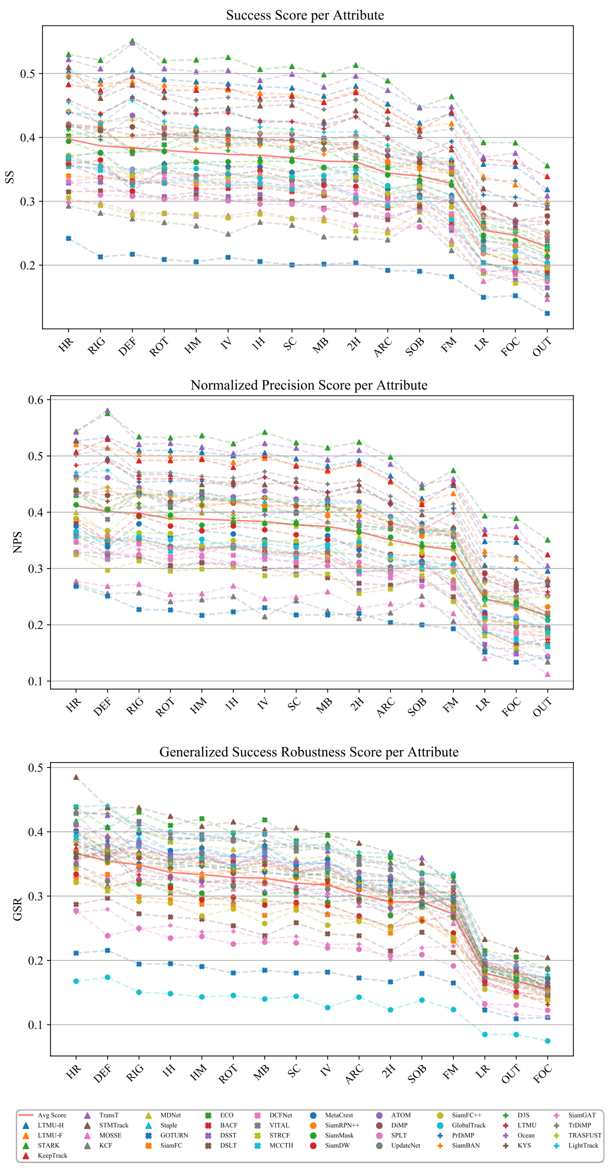


Figure 6.15: SS, NPS, and GSR performance achieved under the MSE protocol of the 40 selected trackers with respect to the sequence attributes available in TREK-150. (The results for the POC attribute are not reported because this attribute is present in every sequence). The red plain line highlights the average tracker performance.

for the case of deformable objects (DEF) and the presence of similar objects (SOB). In the latter situations, the performance of the two trackers is around the same. For the NPS, STARK results better than TransT in general, even though the gap between them is reduced. TransT outputs better centered bounding boxes in the DEF and SOB conditions. Considering the GSR measure, we observe that STMTrack results in the best methodology across most of the attributes. The improvement over the other solutions is particularly significant in the presence of the challenging conditions of small objects (LR), target out-of-view (OUT), and full occlusion (FOC). STMTrack exhibits also a much better score with objects appearing in large size (HR). The ECO tracker instead provides longer references to targets in the case of head motion (HM), motion blur (MB), and fast motion (FM). With respect to the introduced FPV trackers, we have that the performance of LTMU is improved by LTMU-H and LTMU-F overall. In particular, LTMU-H gives the largest improvements in the presence of small objects (LR), target out-of-view (OUT) and full occlusion (FOC). These outcomes tell that the introduced FPV-specific components are particularly helpful in the circumstances that affect the trackers the most.

6.5.6 Action Analysis

The plot in Figure 6.16 reports the MSE protocol results of SS, NPS, and GSR with respect to the action verb labels associated to the actions performed by the camera wearer in each video sequence. In general, we observe that the actions mainly causing a spatial displacement of the target (e.g. “move”, “store”, “check”) have less impact on the performance of the trackers. Instead, actions that change the state, shape, or aspect ratio of the target object (e.g. “remove”, “squeeze”, “cut”, “attach”) generate harder tracking scenarios. Also the sequences characterized by the “wash” action verb lead trackers to poor performance. Indeed, such an action makes the object harder to track because of the many occlusions caused by the persistent and severe manipulation washing involves. It can be noted from the plots that no tracker prevails overall, but STARK and TransT occupy the top stops especially in the plots relative to SS and NPS. In general, the performance of the trackers varies much across the different actions showing that various approaches are suitable to track under the different conditions generated.

The plots in Figure 6.17 presents the performance scores of the trackers with respect to the target noun labels, i.e. the categories of target object. Rigid, regular-sized objects such as “pan”, “kettle”, “bowl”, “plate”, and “bottle” are among the ones associated with higher average SS greater or around 0.5, but some of them (e.g. “plate” and “bottle”) lead to lower GSR scores meaning that trackers provide a spatially accurate but short temporal reference to such kind of objects. In contrast, other rigid objects such as “knife”, “spoon”, “fork” and “can” are more difficult to track from the point of view of all the considered measures (the scores are around 0.3 or lower). This is probably due to the particularly thin shape of this kind of objects and the light reflectance they are easily subject to. Deformable objects such as “sponge”, “onion”, “cloth” and “rubbish” are in general also difficult to track.

6.5.7 FPV Downstream Task Evaluation

Table 6.8 presents the results of the evaluation of all the 40 trackers in relation to the FPV downstream task described in Section 6.4.1. Despite we are showing that FPV introduces challenges for current trackers, with this experiment we want to assess

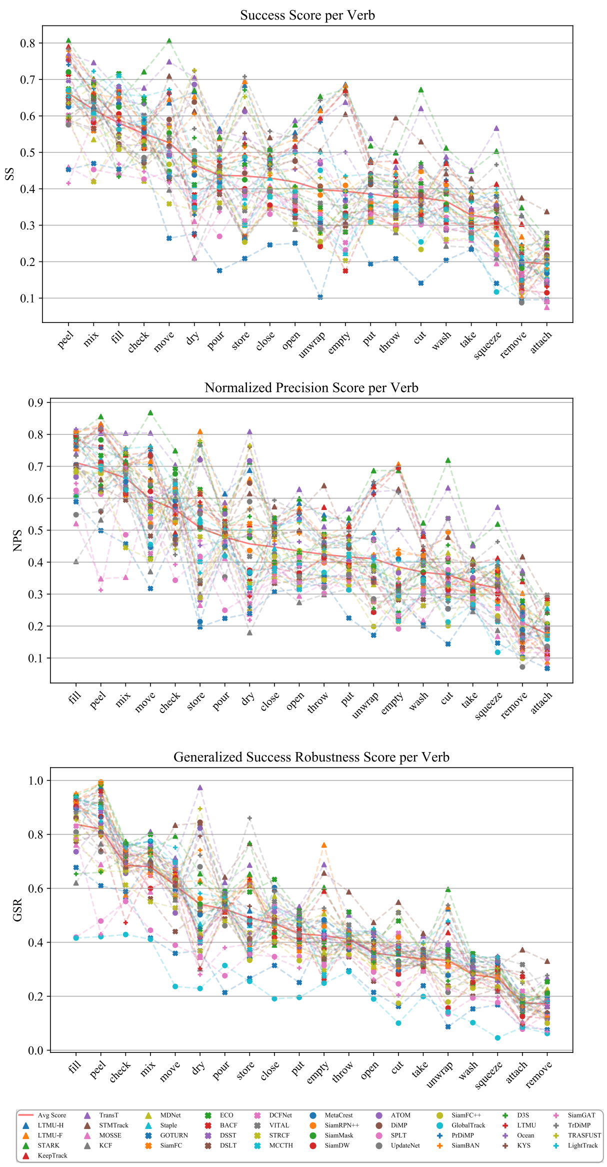


Figure 6.16: SS, NPS, and GSR performance achieved under the MSE protocol of the 40 selected trackers with respect to the action verbs performed by the camera wearer and available in TREK-150. (The results for the POC attribute are not reported because this attribute is present in every sequence). The red plain line highlights the average performance.

whether they can be still exploited in the FPV domain to obtain information about the objects' locations and movements in the scene [180, 221, 182, 202, 224]. The results

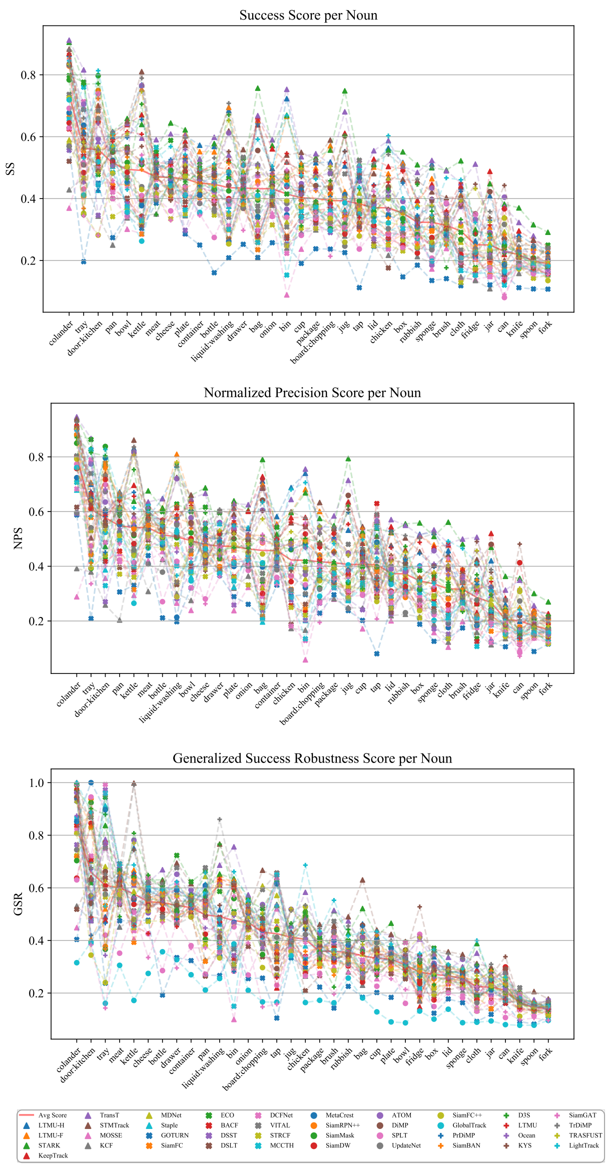


Figure 6.17: SS, NPS, and GSR performance achieved under the MSE protocol of the 40 selected trackers with respect to the target noun categories available in TREK-150. (The results for the POC attribute are not reported because this attribute is present in every sequence). The red plain line highlights the average tracker performance.

given in the first four columns of the table report the Recall of the proposed hand-object video detection pipeline in which each tracker is included, as well the SS, NPS, and GSR results achieved by the tracker in an OPE-like fashion on the same sub-sequences on which the pipeline is run. The outcomes show that, if initialized with a proper bounding box for the object involved in the interaction, the trackers are able to maintain the spatial and temporal reference to such an object for all the interaction period with promising accuracy. Indeed, the Recall values achieved by the proposed hand-object interaction system almost reaches 0.700. By comparing the Recall with the tracker performance scores (SS, NPS, GSR), it can be noted that there is a correlation between the first and the SS, since the ranking of the trackers according to the first measure is very similar to the one of the second measure. It is also worth observing that the SS, NPS, GSR scores achieved with this protocol reflect the performance achieved by the trackers with the OPE protocol on the full sequences of TREK-150, as reported in Figure 6.10. These results report that evaluation of the trackers' performance on the original sequences of TREK-150 can lead to conclusions about the behavior of the trackers in particular application scenarios. Furthermore, the reader might wonder why there is such a large absolute difference in the values of the SS, NPS, and GSR present in Table 6.8 and those in the brackets of Figure 6.10. This can be explained by the fact that in the considered downstream FPV task evaluation the lengths of the video sequences are very short (the average length is of 55 frames). In contrast, the average length of the full video sequences present in TREK-150 is 649 frames, which is are much higher than the previously discussed number. Such a shorter duration of the videos simplifies the job of the trackers since the variations of the target object and the scene are less significant in these conditions rather than in longer sequences. A justification to this explanation is also given by the GSR results of Figure 6.10. For example, on such measure, STARK achieves 0.395 which means that such an algorithm tracks successfully until the 39.5% of a sequence length. In number of frames, such a fraction is 256 on average. This value is much higher than the length of the sub-sequences and it explains why the performance of STARK is so successful in the context of this FPV application.

The final four columns of Table 6.8, report the Recall of the hand-object detection pipeline as well as the trackers' SS, NPS, and GSR, achieved when considering the initialization of the trackers by the bounding box given by the HiC detector [224]. With respect to the values in the first four columns, the results are much lower. This is due to the performance of HiC which struggles to find a valid hand-object detection in the proposed video-based pipeline. This issue delays the initialization of the tracker making the overall pipeline not detecting and localizing the hand-object interaction in many frames. Overall, in this setting we observe that the ranking of the trackers changes. Trackers that do not achieve the top spots in the ground-truth-based experiments (e.g. KeepTrack, LTMU-H) now compete in making the hand-object interaction system more accurate (i.e. they increase the Recall). Considering that in this situation the initialization box is not as accurate as the ground-truth, such an outcome suggests that the different trackers are subject to the noise in the initialization in a different manner. Particularly, it results that TransT is a better suited methodology for tracking objects starting from an initialization given by an object detection algorithm. Similarly as before, the SS is the score that better correlates with the Recall of the overall pipeline.

6.5.8 Contribution of Trackers to FPV Tasks

To understand if the employment of trackers brings advantages with respect to the more standard object localization solutions used in FPV [224, 179], we compared the Recall

Table 6.8: Results of the experiment in which trackers are evaluated by the Recall of an FPV hand-object interaction detection pipeline where trackers are used as localization method for the object involved in the interaction. The firsts of the two blocks of columns present the results of the proposed system in which each tracker is initialized with: the ground-truth object localization in the first frame of the hand-object interaction; the bounding box given by HiC in its first valid hand-object interaction detection. The last three columns report the SS, NPS, and GSR results achieved by each tracker with the OPE protocol on the sub-sequences yielded by the hand-object interaction labels. Best results, per measure, are highlighted in **red**, second-best in **blue**, third-best in **green**.

Tracker	Initialization detection							
	Ground-truth				HiC			
	Recall	SS	NPS	GSR	Recall	SS	NPS	GSR
STARK	**0.680**	**0.600**	**0.625**	0.606	0.152	**0.141**	**0.145**	0.157
TransT	**0.660**	**0.593**	**0.622**	**0.628**	**0.169**	**0.152**	**0.162**	**0.158**
TrDiMP	**0.623**	**0.564**	**0.580**	**0.646**	**0.157**	**0.140**	0.144	0.157
STMTrack	0.615	0.554	0.565	**0.668**	0.147	0.138	0.143	**0.171**
KeepTrack	0.612	0.551	0.575	0.595	**0.158**	**0.140**	**0.148**	0.149
LTMU-H	0.600	0.541	0.573	0.593	**0.157**	**0.140**	**0.148**	0.155
LTMU-F	0.587	0.534	0.562	0.585	0.149	0.132	0.139	0.155
LTMU	0.573	0.519	0.544	0.561	0.146	0.132	0.144	0.144
PrDiMP	0.562	0.528	0.545	0.574	0.139	0.127	0.132	0.149
LightTrack	0.561	0.523	0.547	0.620	0.145	0.136	0.142	**0.160**
D3S	0.554	0.522	0.546	0.582	0.126	0.126	0.140	0.148
ECO	0.539	0.506	0.545	0.625	0.137	0.118	0.123	0.154
Ocean	0.522	0.512	0.540	0.573	0.124	0.120	0.119	0.145
KYS	0.519	0.505	0.547	0.592	0.120	0.118	0.125	0.146
SiamRPN++	0.513	0.478	0.481	0.568	0.134	0.121	0.132	0.141
SiamBAN	0.517	0.479	0.499	0.571	0.135	0.122	0.136	0.145
TRASFUST	0.509	0.500	0.520	0.592	0.132	0.128	0.132	0.150
ATOM	0.507	0.497	0.519	0.599	0.130	0.126	0.129	0.154
VITAL	0.496	0.489	0.508	0.606	0.122	0.116	0.126	0.155
SiamMask	0.488	0.460	0.479	0.549	0.133	0.118	0.131	0.147
DiMP	0.483	0.488	0.519	0.586	0.120	0.120	0.127	0.144
DCFNet	0.482	0.473	0.513	0.589	0.118	0.111	0.125	0.146
MetaCrest	0.481	0.475	0.502	0.597	0.115	0.109	0.115	0.145
Staple	0.440	0.439	0.460	0.556	0.110	0.104	0.111	0.137
MCCTH	0.437	0.436	0.457	0.555	0.110	0.100	0.108	0.132
DSLT	0.432	0.432	0.462	0.513	0.107	0.105	0.111	0.129
BACF	0.429	0.443	0.478	0.569	0.109	0.103	0.114	0.138
SiamFC++	0.429	0.429	0.444	0.532	0.114	0.108	0.116	0.138
GlobalTrack	0.424	0.394	0.388	0.368	0.102	0.097	0.103	0.093
SiamDW	0.419	0.432	0.455	0.551	0.115	0.110	0.121	0.143
UpdateNet	0.419	0.414	0.410	0.520	0.110	0.100	0.106	0.132
MDNet	0.418	0.437	0.465	0.555	0.098	0.106	0.113	0.141
SiamFC	0.411	0.425	0.445	0.523	0.112	0.105	0.111	0.136
DSST	0.405	0.433	0.444	0.560	0.105	0.099	0.104	0.132
STRCF	0.403	0.425	0.449	0.556	0.097	0.100	0.107	0.131
SiamGAT	0.403	0.421	0.409	0.455	0.091	0.101	0.104	0.116
KCF	0.393	0.419	0.430	0.551	0.096	0.095	0.103	0.129
SPLT	0.381	0.402	0.403	0.498	0.093	0.095	0.099	0.120
MOSSE	0.348	0.393	0.383	0.522	0.090	0.096	0.100	0.134
GOTURN	0.348	0.371	0.387	0.493	0.114	0.104	0.112	0.138

results of the trackers presented in Table 6.8 with the Recall results of the original hand-object interaction detector HiC [224] which processes the frames independently. This solution achieves a Recall of 0.080 which results very low when compared to the 0.169, 0.158, and 0.157 achieved by the pipelines exploiting TransT, KeepTrack, TrDiMP, LTMU-H respectively.

In addition to this experiment, we evaluated the performance of a Faster R-CNN [235] instance trained on EK-55 [179] when used as a naive tracking baseline. On the full sequences of TREK-150 such a solution achieves an OPE-based SS, NPS, and GSR of 0.323, 0.369, 0.044 respectively, and runs at 1 FPS. Comparing these results with the ones presented in Figure 6.10, we clearly notice that trackers, if properly initialized by a detection module, can deliver faster, more accurate, and much temporally longer object localization than detectors.

Overall, these outcomes demonstrate that visual object trackers can bring benefits to FPV application pipelines. In addition to the ability of maintaining reference to specific object instances, the advantages of tracking are achieved in terms of better object localization and efficiency. We hence expect that trackers will likely gain more importance in FPV as new methodologies explicitly considering the first person point of view are investigated.

6.6 Conclusions

In this chapter, we studied a new computer vision domain from the visual tracking point of view. Particularly, we presented the first systematic evaluation of visual object tracking in first person vision (FPV). The analysis has been conducted with standard and novel measures on the newly introduced TREK-150 benchmark, which contains 150 video sequences extracted from the EK [179, 203] FPV dataset. TREK-150 has been densely annotated with 97K bounding-boxes, 17 sequence attributes, 20 action verb attributes, and 34 target object attributes, as well as with spatial annotations for the camera wearer's hands and their state of interaction with the target object. The performance of 38 state-of-the-art generic-object visual trackers and two baseline FPV trackers was analysed extensively on the proposed dataset. The investigation has conducted to the following conclusions. The performance of all the benchmarked trackers, both deep learning-based and non, is decreased when compared with the respective accuracy on other popular visual object tracking benchmarks. This is explained by the different nature of images and the particular characteristics introduced by FPV which offer new and challenging conditions. The analysis revealed that deep learning-based trackers employing online adaptation techniques achieve better performance than the trackers based on siamese neural networks or on handcrafted features. The introduction of FPV-specific object localization modules in a tracking pipeline increased the performance of the pipeline, demonstrating that particular cues about the domain influence the tracking accuracy. The performance of the trackers was then studied in relation to specific attributes characterising the visual appearance of the target and the scene. It turned out that the most challenging factors for trackers are the target's out of view, its full occlusions, its low resolution, as well the presence of similar objects or of fast motion in the scene. Trackers were also analyzed based on the action performed by the camera wearer as well as the object category the target belongs to. It resulted that actions causing the change of state, shape, or aspect ratio of the target affected the trackers more than the actions causing only spatial changes. We observed that rigid thin-shaped objects are among the hardest ones to track. Finally, we evaluated the trackers in the context of the FPV-specific application of video-based hand-object interaction detection. We included each tracker in a pipeline to tackle such a problem, and evaluated the performance of the system to quantify the tracker's contribution. We observed that the trackers demonstrate a behavior that is consistent with their overall performance on the sequences of TREK-150. Even though FPV introduced challenging factors for trackers, the results in such a specific task demonstrated that current trackers can be used successfully if the video sequences in which tracking is required are not too long. We also demonstrated that trackers bring advantages in terms of object referral and localization, and efficiency, over object detection. In conclusion, we believe that there is potential in improving FPV pipelines by employing visual trackers as well as there is room for the improvement of the performance of visual object trackers in this new domain.

7 From Tracking with Bounding-Boxes to Tracking with Segmentation Masks

The ideas presented in the previous chapters are all based on the assumption that the state of the targets is represented as a bounding-box. This has been a common premise in most of the successful tracking methodologies developed until today, such as mean shift algorithms [25], part-based methods [29, 30], SVM learning [208], correlation filters [31, 32, 33, 34, 6]. A lot of effort has been also spent to build robust trackers based on convolutional neural networks (CNNs) to work with such kind of state representation. Among the many different approaches exploiting such deep learning techniques [41, 42, 40, 236, 50, 53, 54], CNN-based discriminative correlation filters [59, 60, 61, 62] and siamese CNNs [43, 44, 45, 86, 46, 47] emerged as the year-by-year bar-raising methodologies. Trackers based on such principles showed a remarkable performance across all the available tracking benchmarks [39, 38, 71, 70, 37, 73, 92], almost reaching a 70% of bounding-box overlap accuracy. Such an high and generalized performance poses the question about whether the bounding-box based measures have been now saturated. Moreover, the object detection community proved that humans hardly distinguish a bounding-box prediction that has a 30% overlap from one with 50% [237]. In light of this and considering that would be even more challenging to distinguish between a 50%-overlap box from a 70% one, it is fair to ask ourselves if a tracking system with 100% overlap accuracy is really necessary for applications. From such considerations, one could wonder if the time is done for bounding-box representations. Furthermore, starting from 2020, the major visual tracking communities (VOT2020[1] and MOT[2]) raised the bar in their annual challenges by requesting trackers binary segmentation maps as target state representation. Segmentation representations are not new in the visual tracking panorama. In many applications, model-based al-

[1] `https://votchallenge.net/vot2020/`
[2] `https://motchallenge.net`

gorithms used contours [238, 239] or masks [240, 241] for tracking particular objects. From a more general point of view, the recent video object segmentation (VOS) problem requires to produce the segmentation masks of generic target objects in a video, given the mask of each in the first frame. The currently available solutions propose highly accurate methods in terms of segmentation ability [242, 243, 244, 245, 246, 247], but with the drawback of poor robustness and speed performance. This is due to characteristics of the available benchmarks in the VOS community [248, 249, 250], that do not include challenging situations from a tracking point of view. In fact, such datasets provide temporally short sequences where the target covers a large fraction of the space of the frames, its appearance does not suffer major changes, and very few background distractors are present. The performance of such methods on standard VOT benchmarks was proven very poor [73, 119] and to mitigate such behavior, the SiamMask [47] and D3S [124] algorithms have been proposed recently. These solutions adapted, respectively, the siamese correlation approach and discriminative correlation filters to segmentation outputs, and showed promising results while performing in real-time.

In this chapter, we overcome the bounding-box representation and a provide a way to easily develop trackers able to output segmentation masks. Our belief is that the huge effort spent by the tracking community in developing bounding-box based trackers should not be ignored and that it can be still exploited even in the segmentation-based tracking domain. With such an idea in mind, we propose to explore what is currently available in the computer vision literature that can be adapted to make any bounding-box tracker output segmentation masks. In particular, we propose to extensively evaluate three deep learning methods to generate segmentation masks after bounding-boxes: Box2Seg [247], SiamMask [47], and AMP [251]. Two were already proposed for this task [247, 251], but their capabilities were not studied in depth. The other is a recent segmentation tracker [47] that we reinterpret as a segmentation module. Our evaluations are based on a framework that requires a bounding-box tracker to provide a coarse localization of the object through bounding-boxes, and then a segmentation module conditioned on the target object is employed to provide its precise localization with segmentation masks. Along with the practical considerations, the chapter will show that this combination can produce segmentation-based trackers able to compete with the recently proposed methods [47, 124] on the VOT2020 [119] and DAVIS [248, 249] benchmarks.

7.1 Related Work

Combining segmentation methods and trackers has been increasingly tackled in the last years. SiamMask [47] and D3S [124] employed a CNN decoder module [252, 253] to refine a latent representation constructed by a cross-correlation operation and discriminative filter, respectively. Zhang *et al.* [254] proposed to use ECO tracker's [60] bounding-box predictions to improve the segmentation performance of the OSVOS [242] VOS method. Similarly, [247] adapted a deep CNN for semantic segmentation to generate a segmentation mask after the bounding-box proposal of a tracking-by-detection approach. The combination of these methods achieved promising results, but they were mainly focused on the VOS task. Additionally, they did not provide any extensive evaluation considering different trackers and segmentation methods. In this study, we aim to provide a deep analysis of such combination on both visual tracking and VOS benchmarks.

A competitive approach to the one introduced in this Chapter has been released after the publication of our study. Particularly, Yan *et al.* proposed AlphaRefine [255], a CNN architecture placed after the tracker's bounding-box predictions that aims to provide a segmentation of the target object. The output of such module is also used to correct the tracker's last known target position for better searching area extraction in the next frame. The proposed CNN is composed of layers – such as the pixel-wise correlation – and output heads particularly designed for precise target localisation during tracking.

7.2 Methodology

In this chapter, we study how state-of-the-art off-the-shelf bounding-box trackers can be augmented to track an object with the requirement of a segmentation representation. Our idea is based on the belief that the much effort spent in developing algorithms to predict the motion of a target is relevant even if a segmentation is required. To implement our analysis, we design a framework where a bounding-box tracker is first used to get a coarse localization of the target object, and then a target-conditioned segmentation method is executed to generate a pixel-wise map. Under this setup, any bounding-box based tracker can be transformed into a segmentation tracker. Considering separately tracking and mask generation carries practical advantages: (i) the performance of a segmentation tracker can be analyzed more consistently, by separating the error committed in the localization from the error in shape definition; (ii) flexibility of easily switch tracking and segmentation modules to adapt to application needs; (iii) availability of two different forms of output (bounding-box and mask) that are obtained with independent modules.

In the following of this section, we first introduce the framework employed for the analysis. Then, an abstract description of each of the selected segmentation methods will be given.

7.2.1 Segmentation Tracking Framework

We first define the key elements of the framework. A video

$$\mathcal{V} = \{F_t\}, t \in \{0, \cdots, T\}, T \in \mathbb{N} \tag{7.1}$$

is considered as a T long sequence of frames $F_t \in \mathcal{I}$, where $\mathcal{I} = \{0, \cdots, 255\}^{W \times H \times 3}$ is the space of RGB images. We treat a bounding-box tracking algorithm as a function

$$\mathbf{T} : \mathcal{I} \to \mathbb{R}^4 \tag{7.2}$$

that is inputted with frame F_t and produces a bounding-box estimate $b_t = [x_t, y_t, w_t, h_t]$ as a real-valued vector containing the center coordinates x_t, y_t, and the width and height w_t, h_t (in the image coordinate system).[3] In a similar fashion, we consider a target-based segmentation algorithm as the function

$$\mathbf{M} : \mathcal{P} \times \mathcal{Z} \to \{0, 1\}^{W' \times H'} \tag{7.3}$$

[3]At $t = 0$, $\mathbf{T}$ is initialized with F_0 and the ground-truth bounding-box $b_0^{(g)}$.

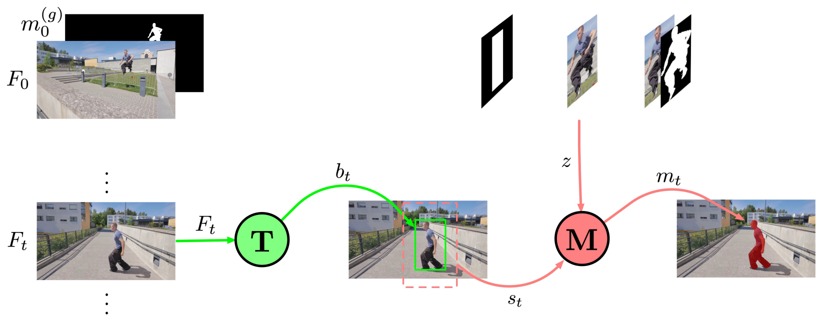


Figure 7.1: Graphical representation of the framework used for the evaluation. $m_0^{(g)}$ outlines the target to be tracked in the first frame F_0 of a video. At every step t, the frame F_t is first given in input the the tracker $\mathbf{T}$ which outputs a bounding-box estimate b_t. This, together with a factor k, is employed to crop a searching area s_t in F_t. s_t is inputted to the mask generation algorithm $\mathbf{M}$ which is conditioned on the target template z computed in different form depending on $\mathbf{M}$'s input requirements. $\mathbf{M}$ returns the segmentation of the target inside s_t. The output mask m_t is finally built by placing $\mathbf{M}$'s output inside a zero-matrix at the location of s_t.

which is given an image patch $s_t \in \mathcal{P} = \{0, \cdots, 255\}^{W' \times H' \times 3} \subseteq \mathcal{I}$ extracted from F_t and a template image $z \in \mathcal{Z}$ of the target object, and outputs a binary segmentation mask with zero-elements belonging to the background and one-elements defining the pixels of the target.

Given these concepts, the segmentation tracking procedure works as follows. At every time step t of a video $\mathcal{V}$, F_t is first given to the tracker $\mathbf{T}$ to produce b_t. Then, F_t and b_t are used to extract a searching area

$$s_t = F_t[x_t, y_t, k \cdot w_t, k \cdot h_t] \tag{7.4}$$

which is the area of F_t localized by the coordinates of b_t and which width and height are scaled by the factor $k \in \mathbb{R}$. s_t and z are given to the segmentation algorithm $\mathbf{M}$ to produce the pixel-wise mask of the target inside s_t.[4] The output mask m_t is finally built by placing $\mathbf{M}$'s output at the s_t location of a zero-matrix with size $W \times H$.

A graphical representation of the described framework is shown in Figure 7.1.

7.2.2 Target-Conditioned Segmentation Methods

In this subsection we describe the target-conditioned segmentation methodologies we analyzed. Three conceptually different approaches were chosen:

- an adapted semantic segmentation network [247, 256], that we name SemSeg;
- a module based on the siamese correlation framework [47], referred as SiamSeg;
- a few-shot segmentation algorithm [251], called FewShotSeg.

[4]The details to obtain z are described in each subsection describing the segmentation methods.

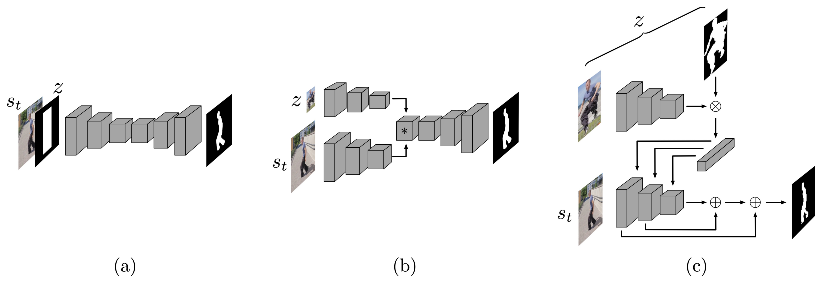


Figure 7.2: Visual representation of the methodologies employed to generate target segmentations. (a) shows SemSeg, a deep segmentation model adapted to take in input a 4-channel tensor composed of s_t and z and produce the mask of the object. (b) presents SiamSeg, the siamese framework where a cross correlation operation between z and s_t features is employed to first locate the object, and then to produce its segmentation mask. (c) shows FewShotSeg, a few-shot segmentation algorithm that is adapted for visual segmentation tracking, by considering z as the support set and s_t as the query image.

SemSeg. The first target-conditioned segmentation method we analyzed was proposed as Box2Seg in [245, 247]. The idea is to adapt a state-of-the-art fully convolutional deep neural network for image segmentation to target segmentation. Given an RGB image and an additional input channel containing coarse information regarding the position of the target, this module produces a detailed segmentation of the latter. In the context of our framework, the RGB channels of the searching area s_t are concatenated with the template channel

$$z = \{0,1\}^{k\cdot w_t, k\cdot h_t} \tag{7.5}$$

which is a binary mask of the same size of s_t, and which positive elements are located inside the area defined by b_t. z is computed at every time step t, and the 4-channel input resulting from the concatenation is given to the network which produces the segmentation of the target inside the searching area. A visualization of this approach is proposed in Figure 7.2 (a). The network is trained offline by exploiting object segmentation, instance segmentation, and/or VOS datasets. The training pairs are formed as batches of inputs-targets, where the first are composed by searching area and template (built using the bounding-box that encloses the ground-truth segmentation), and the second are the actual object masks. Optimization is done by solving a two class segmentation problem (foreground-background) defined as the minimization of a pixel-wise classification loss (cross-entropy, Dice loss, etc.). This approach has the advantage of requiring just the bounding-box as first-frame target definition.

SiamSeg. As second mask generation method, we reinterpreted the siamese correlation framework for segmentation tracking [47]. The general view of this scheme is to first locate the target template in the higher-level feature space of template and searching area, and then project the localization into the segmentation space. These steps are jointly implemented with an encoder-decoder CNN architecture, which capabilities are acquired through an end-to-end offline procedure in which the whole model is optimized by minimizing a foreground-background pixel-wise classification loss. The training examples are pairs of searching area-template inputs, and ground-truth target masks, where searching area and template are sampled without temporal correlation.

Following this intuition, we adapted such method in our framework as follows. The target template is the image crop

$$z = F_0[x_0, y_0, w_0, h_0] \tag{7.6}$$

extracted from the first frame F_0 of the video, using the ground-truth bounding-box $b_0^{(g)} = [x_0, y_0, w_0, h_0]$. $b_0^{(g)}$ is obtained as the box that encloses the ground-truth mask $m_0^{(g)}$. The features $\widehat{z}$ of z are computed with a forward pass through the encoder module just at $t = 0$. At every other t, $\widehat{z}$ is cross-correlated to the encoded representation $\widehat{s_t}$, and the resulting activation map is then refined by the decoder module, and ultimately placed into m_t. The procedure is depicted in Figure 7.2 (b) and, as for SemSeg, it just requires the target definition as a bounding-box.

FewShotSeg. The last analyzed methodology treats target-conditioned segmentation as a few-shot segmentation problem. In such a setting, the goal is to provide a pixel-wise segmentation of a target object inside a query image, given a so-called support-set, i.e. one or more (few-shot) image and mask examples of the target. Algorithms for this problem are generally designed as fully CNNs, where the segmentation ability is guided by other convolutional branches or by model parameters that are made dependent on the support-set.

This view of few-shot segmentation can be reframed for the purpose of segmentation tracking. In our setting, the support-set is considered as the target template

$$z = (F_0[x_0, y_0, w_0, h_0], m_0^{(g)}[x_0, y_0, w_0, h_0]) \tag{7.7}$$

that is the pair of the image crop that contains the visual appearance of the target in F_0, and the relative cropped ground-truth mask. The crops are constructed considering $b_0^{(g)} = [x_0, y_0, w_0, h_0]$. The searching area s_t is extracted after every b_t of $\mathbf{T}$ and it is considered as the query image. Together with the template (the support-set), they are given to the few-shot segmentation model to produce the target segmentation. A graphical example of this methodology is proposed in Figure 7.2 (c). With respect to the previous methods, employing FewShotSeg requires the definition of the target object through a mask.

7.3 Experimental Setup

In this section, we report the experimental procedures we performed to implement and analyze the previously presented methodologies. All experiments were run on a machine with an Intel Xeon E5-2690 v4 @ 2.60GHz CPU, 320 GB of RAM, and an NVIDIA TITAN V GPU. Code for tracker and segmentation methods was implemented in Python.

7.3.1 Trackers

The trackers selected for the analysis were KCF [32], DCFNet [230], MDNet [40], MetaCrest [132], SiamFC [43], SiamRPN [44], ECO [60], ATOM [61], and DiMP [62]. Such algorithms were chosen because they tackle visual tracking by different approaches and so can provide performance of various quality. For each of them, we used the public code made available by the authors. We tried the best to respect default parameters and settings.

7.3.2 Segmentation Modules

SemSeg. To implement this methodology, we followed the details of the Box2Seg refinement module provided in [247, 245]. The DeepLab-v3 architecture [256] for image segmentation was translated for the task of interest. ImageNet [19] pre-trained ResNet-50 [103] was employed as backbone network and adapted to receive the 4-channel tensor. Before being inputted, the concatenated RGB and template channels were resized to 385×385 pixels. During training, the searching area was enlarged by the factor k, chosen uniformly in $\{1, 1.25, 1.5, 1.75, 2\}$. Batches of 12 input-target mask pairs were sampled from a training set composed of the training sets of COCO [257], YouTube-VOS [250], and DAVIS 2016 and 2017 [248, 249]. Learning rate was set to 10^{-5} for the backbone layers, and to 10^{-4} for all the others. Training was carried on until the mIoU [258], computed over the foreground and background classes, stopped improving on a custom validation set composed of the validation sets of the aforementioned datasets.

SiamSeg. The second approach introduced in subsection 7.2.2 was implemented through the segmentation tracker SiamMask [47]. We used the code provided by the authors along with the pre-trained models. For completion, we present to the reader some information about the training procedures performed by the authors. The SiamMask architecture model was trained in two-stages: first, the encoder module based on ResNet-50 [103] was trained for target localization by optimizing a multi-task loss for similarity maximization and RPN [235] detection. After that, the decoder module designed as [253] was attached to the intermediate cross-correlation map and trained by minimizing a foreground-background pixel-wise cross-entropy loss. The training set used was a combination of ImageNet-VID [19], COCO [257] and YouTube-VOS [250]. Before being inputted to the model, z and s_t were resized to 127×127 and 255×255 pixels respectively.

FewShotSeg. As a few-shot segmentation module, we employed the strategy proposed in [251], which is a recently introduced state-of-the-art method that has been shown to perform well also in VOS tasks. The authors proposed a sample efficient method to segment an unseen class object via a multi-resolution imprinting procedure of adaptive masked proxies (AMP). AMPs are constructed by a Normalized Masked Average Pooling (NMAP) operation between the CNN embeddings of the support set's RGB sample and its relative binary mask. The AMP representations are used to imprint [259], at multiple resolutions, the CNN embeddings computed on the query image. The VGG16 [260] architecture is employed as a backbone feature extractor, and skip connections

Table 7.1: Results of the baseline experiment on VOT2020. Best segmentation method results, per tracker, are highlighted in red (Rectangular Mask results are excluded).

Tracker	SemSeg			SiamSeg			FewShotSeg			Rectangular Mask		
	$EAO_{\uparrow}$	$A_{\uparrow}$	$R_{\uparrow}$	$EAO_{\uparrow}$	$A_{\uparrow}$	$R_{\uparrow}$	$EAO_{\uparrow}$	$A_{\uparrow}$	$R_{\uparrow}$	$EAO_{\uparrow}$	$A_{\uparrow}$	$R_{\uparrow}$
DCFNet	0.203	0.616	0.426	**0.310**	**0.676**	0.558	0.230	0.491	**0.567**	0.184	0.441	0.523
KCF	0.199	0.648	0.371	**0.285**	**0.659**	**0.501**	0.200	0.459	0.499	0.155	0.402	0.432
SiamFC	0.218	0.602	0.446	**0.309**	**0.682**	**0.571**	0.228	0.491	0.563	0.183	0.418	0.537
MetaCrest	0.240	0.602	0.513	**0.336**	**0.657**	0.624	0.250	0.479	**0.647**	0.189	0.390	0.587
SiamRPN	0.356	0.692	0.639	**0.369**	**0.701**	0.651	0.311	0.551	**0.677**	0.247	0.452	0.663
MDNet	0.295	0.638	0.609	**0.371**	**0.662**	0.689	0.308	0.546	**0.723**	0.234	0.440	0.687
ATOM	0.402	0.678	**0.735**	**0.406**	**0.691**	0.723	0.337	0.560	0.731	0.277	0.467	0.738
DiMP	0.410	0.675	0.744	**0.410**	**0.691**	0.730	0.347	0.556	**0.749**	0.278	0.464	0.733
ECO	0.322	0.632	0.735	**0.414**	**0.694**	0.729	0.349	0.561	**0.759**	0.275	0.459	0.746
b-oracle	**0.806**	**0.809**	**0.996**	0.697	0.744	0.970	0.541	0.623	0.941	0.516	0.519	1.0

are also exploited as done similarly in FCN8s [261]. Data extracted from the PASCAL-VOC dataset [258] was used to compose training samples as query image, support-set image, support-set mask, and target mask. Optimization was performed by minimizing the pixel-wise cross-entropy loss between predicted and ground-truth masks. Code and pre-trained model provided by the authors were adapted to our implementation needs.

7.3.3 Benchmarks and Performance Measures

We performed analysis on the VOT2020 benchmark, and the validation sets of the DAVIS 2016 [248] and DAVIS 2017 [249] VOS benchmarks. All provide segmentations as target representations.

For VOT2020 we employed the newly introduced protocol.[5] The novel baseline protocol requires running a tracker on shorter sequences determined by predefined points (anchors). From such starting points, the tracker is initialized with the ground-truth mask and run either forward or backward, depending on the longest sub-sequence yielded by the two directions. The new accuracy ($A_{\uparrow}$) measures the average pixel-wise intersection-over-union between predicted and ground-truth masks, for frames where the tracker did not fail (i.e. the accuracy did not decrease after a certain threshold). The new robustness ($R_{\uparrow}$) expresses the normalized average number of frames where the algorithm successfully tracked the target before drifting. The two measures are joined in a refreshed single performance score known as expected average overlap ($EAO_{\uparrow}$). Version 0.4.2 of the Python toolkit was used to obtain the results.

The protocol used for DAVIS datasets is similar to the One-Pass evaluation (OPE) employed in OTB [71] benchmarks: the tracker is initialized with the mask of the target object in the first frame, and then it is run until the end of the sequence. Performance is measured in terms of the Jaccard index $\mathcal{J}$ which measures the pixel-wise intersection-over-union between the predicted and ground-truth masks. Along with this index, the F-measure $\mathcal{F}$ is employed to evaluate contour accuracy. For both measures, mean ($\mathcal{J}_{\mathcal{M}\uparrow}, \mathcal{F}_{\mathcal{M}\uparrow}$), recall ($\mathcal{J}_{\mathcal{R}\uparrow}, \mathcal{F}_{\mathcal{R}\uparrow}$), and decay ($\mathcal{J}_{\mathcal{D}\downarrow}, \mathcal{F}_{\mathcal{D}\downarrow}$) values are reported. For DAVIS 2017, where multiple objects must be tracked and segmented, we run the trackers independently for each object and then fuse the prediction masks by assigning each pixel to the object that received higher confidence in that location.

[5] `https://data.votchallenge.net/vot2020/vot-2020-protocol.pdf`

Table 7.2: $\mathcal{J}$ results on DAVIS 2016 validation set. Best segmentation method results, per tracker, are highlighted in red (Rectangular Mask results are excluded).

Tracker	SemSeg			SiamSeg			FewShotSeg			Rectangular Mask		
	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$
KCF	0.527	0.570	0.174	0.557	0.616	0.199	**0.580**	**0.688**	**0.162**	0.302	0.200	0.153
DCFNet	0.531	0.574	0.209	0.551	0.627	0.229	**0.564**	**0.674**	**0.178**	0.313	0.183	0.130
MetaCrest	0.574	0.624	0.169	0.595	0.672	0.145	**0.598**	**0.712**	**0.136**	0.323	0.151	0.108
MDNet	0.582	0.635	0.177	0.593	0.656	0.196	**0.610**	**0.717**	**0.143**	0.342	0.198	0.149
SiamFC	0.607	0.661	**0.159**	0.611	0.694	0.177	**0.621**	**0.738**	0.163	0.356	0.234	0.140
ECO	0.615	0.679	**0.099**	0.623	0.744	0.108	**0.626**	**0.748**	0.113	0.375	0.243	0.070
SiamRPN	**0.689**	0.772	**0.089**	0.663	0.782	0.111	0.681	**0.859**	**0.089**	0.417	0.340	0.066
ATOM	**0.723**	**0.846**	**0.074**	0.658	0.785	0.105	0.669	0.845	0.081	0.415	0.345	0.053
DiMP	**0.723**	0.827	**0.086**	0.704	0.844	0.100	0.699	**0.886**	0.095	0.443	0.379	0.027
b-oracle	**0.812**	0.920	**0.020**	0.732	0.896	0.044	0.739	**0.946**	0.052	0.455	0.418	0.008

Table 7.3: $\mathcal{F}$ results on DAVIS 2016 validation set. Best segmentation method results, per tracker, are highlighted in red (Rectangular Mask results are excluded).

Tracker	SemSeg			SiamSeg			FewShotSeg			Rectangular Mask		
	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$
DCFNet	**0.553**	0.587	0.210	0.536	0.596	0.187	0.530	**0.599**	**0.158**	0.155	0.017	0.068
KCF	**0.559**	0.577	0.180	0.525	0.572	0.190	0.542	**0.598**	**0.149**	0.136	0.018	0.119
MetaCrest	**0.599**	0.632	0.193	0.561	**0.637**	**0.136**	0.572	0.634	0.137	0.139	0.019	0.063
MDNet	**0.603**	0.623	**0.170**	0.570	0.616	0.197	0.582	**0.635**	**0.170**	0.163	0.050	0.112
SiamFC	**0.633**	0.665	**0.152**	0.592	0.663	0.159	0.597	**0.675**	0.157	0.156	0.037	0.126
ECO	**0.637**	**0.696**	**0.097**	0.590	0.692	0.102	0.592	0.673	0.117	0.170	0.020	0.066
SiamRPN	**0.713**	**0.783**	**0.105**	0.629	0.707	0.127	0.642	0.752	**0.105**	0.186	0.059	0.081
ATOM	**0.739**	**0.856**	0.098	0.628	0.697	0.111	0.626	0.751	**0.090**	0.178	0.025	0.060
DiMP	**0.744**	**0.821**	**0.108**	0.658	0.754	0.130	0.657	0.767	0.118	0.191	0.071	0.015
b-oracle	**0.843**	**0.918**	**0.033**	0.693	0.805	0.064	0.717	0.873	0.056	0.219	0.073	0.015

7.4 Results

7.4.1 General Performance

Results on VOT2020 benchmark are presented in Table 7.1. Trackers combined with SiamSeg achieve the best overall performance in $\mathrm{EAO}_{\uparrow}$ and $\mathrm{A}_{\uparrow}$. This is explained by the fact the VOT benchmarks include difficult tracking scenarios for trackers, resulting in lower quality bounding-boxes that affect SemSeg and FewShotSeg. Thanks to its more robust segmentation method, SiamSeg allows to recover (to some extent) from inaccurate b_t estimates and so produce more accurate target segmentations. Interestingly, FewShotSeg is the approach that achieves the highest $\mathrm{R}_{\uparrow}$, showing to be the method less susceptible to failure. For all the methods, employing a better tracker is fundamental to improve the overall performance.

In Figure 7.3 some qualitative examples of the segmentation methods are proposed.[6]

Results on the DAVIS 2016 benchmark are reported in Tables 7.2 and 7.3. More weak trackers like DCFNet, KCF, MDNet, MetaCrest, and SiamFC, benefit of FewShotSeg for pixel-wise accuracy. When more precise bounding-box estimates are provided, through ECO, SiamRPN, ATOM, DiMP, SemSeg allows the best $\mathcal{J}_{\mathcal{M}\uparrow}$ performance. For $\mathcal{J}_{\mathcal{R}\uparrow}$ and $\mathcal{J}_{\mathcal{D}\downarrow}$, FewShotSeg is almost always the best approach. For contour accuracy, SemSeg is generally the best method at $\mathcal{F}_{\mathcal{M}\uparrow}$. Better trackers also benefit the same for $\mathcal{F}_{\mathcal{R}\uparrow}$ and $\mathcal{F}_{\mathcal{D}\downarrow}$. For the others, FewShotSeg gets the best results. SiamSeg is the weakest method on this benchmark, justified by the presence of easy tracking situations that put the

[6]For more, please see `https://youtu.be/SODiKBD84_g`.

Table 7.4: $\mathcal{J}$ results on DAVIS 2017 validation set. Best segmentation method results, per tracker, are highlighted in red (Rectangular Mask results are excluded).

Tracker	SemSeg			SiamSeg			FewShotSeg			Rectangular Mask		
	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{D}\downarrow}$
DCFNet	0.443	0.474	0.299	**0.455**	**0.497**	0.281	0.424	0.434	**0.214**	0.283	0.166	0.176
KCF	0.433	0.464	0.277	**0.461**	**0.517**	0.272	0.425	0.451	**0.209**	0.268	0.167	0.198
MDNet	0.444	0.478	0.284	**0.465**	**0.515**	0.260	0.444	0.493	**0.216**	0.284	0.156	0.168
MetaCrest	0.447	0.468	0.276	**0.468**	**0.518**	0.262	0.426	0.443	**0.178**	0.273	0.145	0.155
SiamFC	0.466	0.499	0.260	**0.468**	**0.523**	0.277	0.431	0.454	**0.225**	0.280	0.176	0.196
ECO	**0.498**	0.556	0.244	0.503	**0.567**	0.222	0.458	0.501	**0.178**	0.310	0.220	0.132
SiamRPN	**0.536**	**0.600**	0.233	0.506	0.578	0.237	0.470	0.518	**0.180**	0.321	0.248	0.141
ATOM	**0.566**	**0.659**	**0.148**	0.544	0.626	0.188	0.488	0.547	0.168	0.321	0.251	0.103
DiMP	**0.583**	**0.671**	**0.148**	0.553	0.639	0.170	0.498	0.555	0.162	0.323	0.251	0.093
b-oracle	**0.762**	**0.891**	**0.0**	0.618	0.738	0.073	0.578	0.694	0.059	0.408	0.340	0.0

Table 7.5: $\mathcal{F}$ results on DAVIS 2017 validation set. Best segmentation method results, per tracker, are highlighted in red (Rectangular Mask results are excluded).

Tracker	SemSeg			SiamSeg			FewShotSeg			Rectangular Mask		
	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{D}\downarrow}$
KCF	**0.517**	0.542	0.307	0.500	0.544	0.287	0.506	**0.565**	**0.266**	0.172	0.035	0.178
DCFNet	**0.532**	0.567	0.322	0.512	0.561	0.284	0.511	**0.572**	**0.237**	0.194	0.049	0.140
MDNet	0.525	0.563	0.288	0.513	0.565	0.270	**0.526**	**0.598**	**0.255**	0.184	0.059	0.170
MetaCrest	**0.545**	**0.593**	0.305	0.520	0.567	0.278	0.521	0.583	**0.241**	0.176	0.042	0.146
SiamFC	**0.556**	**0.611**	0.299	0.523	0.583	0.294	0.524	0.601	**0.267**	0.184	0.064	0.184
ECO	**0.592**	**0.663**	0.255	0.553	0.620	0.236	0.553	0.637	**0.226**	0.214	0.055	0.128
SiamRPN	**0.626**	**0.713**	0.259	0.552	0.628	0.257	0.567	0.670	**0.219**	0.210	0.070	0.150
DiMP	**0.663**	**0.765**	**0.181**	0.591	0.675	0.215	0.584	0.685	0.195	0.206	0.059	0.093
ATOM	**0.640**	**0.751**	**0.195**	0.584	0.664	0.218	0.574	0.666	0.201	0.203	0.053	0.104
b-oracle	**0.829**	**0.945**	**0.017**	0.654	0.779	0.097	0.685	0.847	0.078	0.280	0.116	0.028

focus on providing more accurate target segmentations.

On DAVIS 2017, which results are presented in Tables 7.4 and 7.5, SemSeg is still the best approach to use with stronger bounding-box trackers for $\mathcal{J}_{\mathcal{M}\uparrow}$ and $\mathcal{J}_{\mathcal{R}\uparrow}$. FewShotSeg is the method that achieves the most consistent masks across time. For low-performance tracking algorithms, SiamSeg results to be better than the others in $\mathcal{J}_{\mathcal{M}\uparrow}$ and $\mathcal{J}_{\mathcal{R}\uparrow}$, mitigating the lower tracking performance with its target search strategy and showing the increased difficulty of this benchmark than its previous version. In terms of contour performance, SemSeg is the most appropriate method for $\mathcal{F}_{\mathcal{M}\uparrow}$ and $\mathcal{F}_{\mathcal{R}\uparrow}$ performance. For $\mathcal{F}_{\mathcal{D}\downarrow}$, FewShotSeg results in the best solution. Overall, as for VOT2020, in both DAVIS 2016 and 2017 employing better trackers lets achieve the best performances.

7.4.2 Comparison with a Rectangular Segmentation Tracker

In the last block of columns of Tables 7.1, 7.2, 7.3, 7.4, 7.5, we report the performance of the trackers considering their b_t predictions as segmentation masks, i.e. binary mask where the rectangular area defined by b_t is filled with 1. Overall, all the considered segmentation methods improve those baseline results on all the benchmarks and across all measures. This proves that employing the approaches presented in this chapter lets bounding-box trackers improve their accuracy in terms of precise target definition. SemSeg is the method that achieves generally the best improvement, followed by SiamSeg and FewShotSeg.

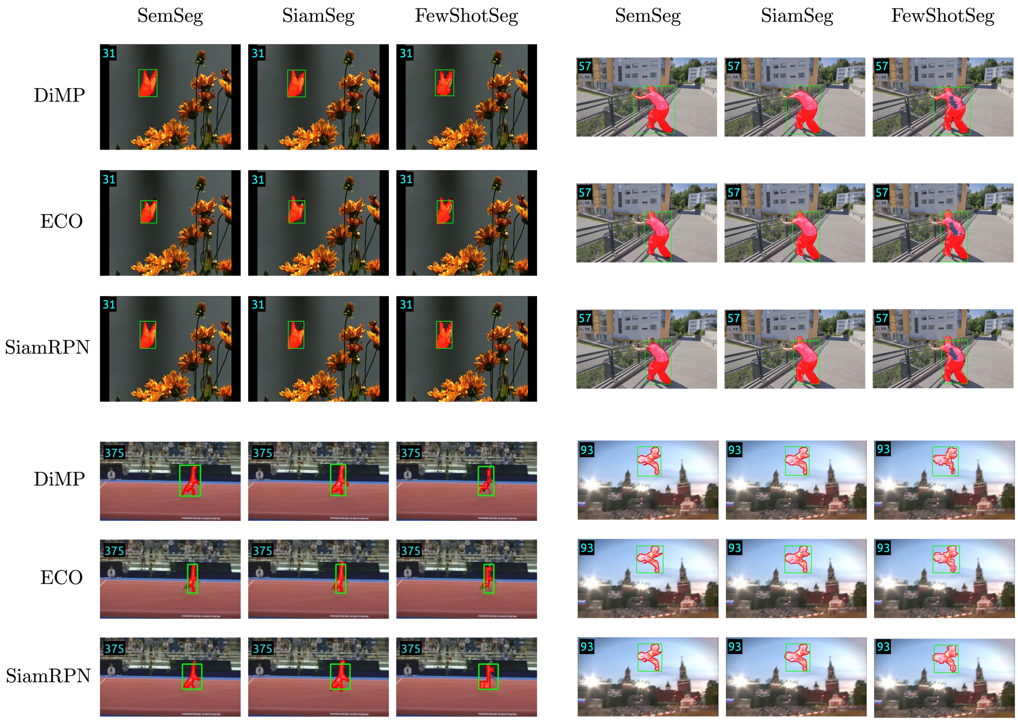


Figure 7.3: Qualitative examples of the segmentation (red superimposed mask) proposed by the three target-conditioned segmentation methods, based on the bounding-box proposals (green rectangles) given by three different trackers.

7.4.3 Comparison with a Bounding-box Oracle Tracker

In the last row of Tables 7.1, 7.2, 7.3, 7.4, 7.5, the performance of a bounding-box oracle based tracker, b-oracle (i.e. the tracker that returns the ground-truth bounding-box $b_t^{(g)}$ at every t), is presented. Given this ground-truth information, SemSeg is the approach that best segments the target object, on every considered benchmark and performance measure. On VOT2020, accuracy and robustness performances reach almost 80% and 100%, meaning that its segmentation capabilities are effective for the objects contained in this dataset. SiamSeg follows with a decrease of 9% and 2.6%, while FewShotSeg shows a much bigger performance loss in $\mathrm{A}_\uparrow$ (-25% than SemSeg) than in $\mathrm{R}_\uparrow$ (-5.5%). FewShotSeg comes after SemSeg in terms of $\mathcal{J}$ and $\mathcal{F}$ on DAVIS 2016, and in terms of $\mathcal{F}$ on DAVIS 2017. SiamSeg gets the weakest performance on DAVIS 2016 but surpasses FewShotSeg in $\mathcal{J}$ on DAVIS 2017.

SemSeg is also the method that suffers the major gap between the b-oracle performance and the best tracker DiMP ($\mathrm{EAO}_\uparrow$ loss -48%, average $\mathcal{J}_{\mathcal{M}\uparrow}$ loss -17.2%, average $\mathcal{F}_{\mathcal{M}\uparrow}$ loss -15.9%). This shows the susceptibility to misaligned bounding-box predictions (we hypothesize this can be mitigated introducing some noise to the input bounding-boxes in the training procedure). The performance decrease happens also for the other

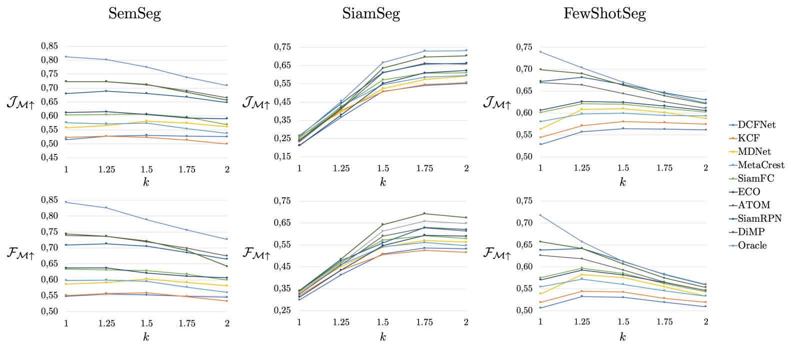


Figure 7.4: Results on the DAVIS 2016 validation set of the sensibility of the target segmentation methods to the size of the searching area. Performance is evaluated in terms of $\mathcal{J}_{\mathcal{M}\uparrow}$ and $\mathcal{F}_{\mathcal{M}\uparrow}$.

methods, although with less magnitude.

7.4.4 Separating Localization and Segmentation Error

The results obtained with b-oracle and the rectangular mask output allow us to determine the tracking and segmentation error committed by $\mathbf{T}$ and $\mathbf{M}$ respectively. The error $e_{\mathbf{T}}$ committed by the tracker is just the performance difference between b-oracle and $\mathbf{T}$, both considered with rectangular mask output. The error $e_{\mathbf{M}}$ of $\mathbf{M}$ can be computed as the performance difference between b-oracle with $\mathbf{M}$ and $\mathbf{T}$ with $\mathbf{M}$ which tracking performance is corrected by summing $e_{\mathbf{T}}$. In this setting, $e_{\mathbf{M}}$ is considered as the distance from $\mathbf{M}$'s maximum achievable performance, that happens when b-oracle is employed as tracker. For example, when MDNet and SemSeg are executed together, the $\mathrm{A}_{\uparrow}$ error $e_{\mathbf{T}}$ is computed as $e_{\mathbf{T}} = 0.519 - 0.440 = 0.079$, while the $e_{\mathbf{M}}$ is obtained as $e_{\mathbf{M}} = 0.809 - (0.638 + 0.079) = 0.092$. So, it results that the highest loss in accuracy is due to the segmentation than to tracking. If DiMP and SiamSeg are considered, we have an $\mathrm{A}_{\uparrow}$ error $e_{\mathbf{T}} = 0.055$ and $e_{\mathbf{T}} = -0.002$ meaning that SiamSeg compensates the tracking error and even improves the performance of the combination.

7.4.5 Impact of the Searching Area Size

We analyzed how sensible the three segmentation methods are to different sizes of the searching area. In particular, the factor k was studied across the values $\{1, 1.25, 1.5, 1.75, 2\}$ on the DAVIS 2016 benchmark, and the results are shown in Figure 7.4. With SemSeg, all the trackers show a slow decrease in $\mathcal{J}_{\mathcal{M}\uparrow}$ and $\mathcal{F}_{\mathcal{M}\uparrow}$ performance by enlarging the searching area. Best performance are obtained with $k = 1$ or $k = 1.25$ (proven also by the b-oracle based tracker). Similar conclusions can be made for FewShotSeg. The highest $\mathcal{J}_{\mathcal{M}\uparrow}$ is achieved with $k = 1.25$. For larger k, the performance of more weak trackers remains constant, while the performance of stronger trackers slightly decreases. $\mathcal{F}_{\mathcal{M}\uparrow}$ tends to decrease for all the trackers. SiamSeg shows the opposite trend. Better results are obtained with larger searching areas. Specifically, best $\mathcal{J}_{\mathcal{M}\uparrow}$

Table 7.6: Results on the speed analysis (in seconds and FPS) of the combined tracker-segmentation methods. The original tracker speeds are reported in the last two columns (times of the employed implementations). The last row shows the average speed of running just the segmentation methods.

Tracker	SemSeg		SiamSeg		FewShotSeg		Tracker speed	
	s	FPS	s	FPS	s	FPS	s	FPS
MDNet	0.628	1.6	0.580	1.7	0.704	1.4	0.550	1.8
MetaCrest	0.181	5.5	0.134	7.5	0.258	3.9	0.109	9.1
ECO	0.126	8	0.081	12.4	0.220	4.6	0.059	17.0
ATOM	0.123	8.1	0.079	12.7	0.198	5.1	0.050	20.0
DiMP	0.109	9.1	0.068	14.7	0.189	5.3	0.038	26.3
SiamRPN	0.026	12.1	0.047	21.4	0.172	5.8	0.026	38.4
KCF	0.070	14.3	0.034	29.5	0.167	6.0	0.013	78.8
SiamFC	0.064	15.6	0.030	33.5	0.157	6.4	0.008	125.3
DCFNet	0.062	16.2	0.026	39.0	0.143	7.0	0.004	227.8
No tracker	0.062	16.3	0.024	42.8	0.151	6.7	-	-

and $\mathcal{F}_{\mathcal{M}\uparrow}$ performance are obtained with $k \geq 1.75$. Weaker trackers have a smaller performance decrease between 1.75 and 1.5 than stronger ones, while for $k < 1.5$ the performance of all the trackers quickly drops. This can be explained by SiamSeg's training methodology, where the objective is set as target localization and segmentation in large image patches.

7.4.6 Speed Analysis

In Table 7.6 an analysis of the speed of the algorithms is presented. The fastest method to produce a segmentation is SiamSeg which runs at 43 FPS. With this method, DCFNet and SiamFC run in real-time (39 and 34 FPS respectively). Stronger trackers like ECO, ATOM, and DiMP, achieve a speed of 12, 13, and 15 FPS respectively. SemSeg runs independently at 16 FPS, and combined with SiamRPN and DiMP allows a speed of 12 and 9 FPS respectively. FewShotSeg is the slowest method and takes around 7 FPS. In this setup, the speed performance is almost completely taken by the segmentation method and best trackers reach a speed of 5-6 FPS.

7.4.7 State-of-the-art Comparison

Comparison with the state-of-the-art is presented in Table 7.7. The VOS methods outperform every studied $\mathbf{T} - \mathbf{M}$ combination on DAVIS 2016 and 2017, but they show poor speed results. DiMP and ATOM with SemSeg perform better than SiamMask in $\mathcal{J}$ on both DAVIS 2016 and 2017. In terms of $\mathcal{F}$ they outperform also D3S. On DAVIS 2017, D3S is improved by DiMP-SemSeg in every measure. On VOT2020, SiamMask is largely beaten by all the best trackers, combined both with SemSeg and SiamSeg. All the trackers using the second method improves SiamMask, showing its limitations in target localization. ECO and SiamSeg reaches an $\text{EAO}_{\uparrow}$ of 0.414, slightly improving DiMP and ATOM. With the same segmentation method, SiamRPN outperforms D3S in $\text{A}_{\uparrow}$, achieving the best 0.701, while maintaining a quasi real-time speed of 21 FPS.

[7]Since we used SiamMask to implement SiamSeg, for fair comparison we report the results of the same implementation used for segmentation tracking, which has slightly worse performance than presented in the original paper.

Table 7.7: State-of-the-art comparison for the best combinations. Best results are highlighted in red, second-best in blue.

Method	DAVIS 2016				DAVIS 2017				VOT2020			FPS
	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	$\mathcal{J}_{\mathcal{M}\uparrow}$	$\mathcal{J}_{\mathcal{R}\uparrow}$	$\mathcal{F}_{\mathcal{M}\uparrow}$	$\mathcal{F}_{\mathcal{R}\uparrow}$	EAO$_\uparrow$	A$_\uparrow$	R$_\uparrow$	
OSMN [246]	0.740	0.876	0.729	0.840	0.525	0.609	0.571	0.661	-	-	-	7
BoLTVOS [247]	0.781	-	0.812	-	**0.684**	-	**0.754**	-	-	-	-	1
OSVOS [242]	0.798	0.936	0.806	**0.926**	0.566	**0.638**	0.639	**0.738**	-	-	-	0.1
FAVOS [262]	0.824	**0.965**	0.795	0.894	0.546	0.611	0.618	0.723	-	-	-	0.8
RGMP [243]	0.815	0.917	0.820	0.908	0.648	-	0.686	-	-	-	-	8
OnAVOS [244]	**0.857**	-	**0.842**	-	0.610	-	0.661	-	-	-	-	0.1
PReMVOS [245]	**0.849**	**0.961**	**0.886**	**0.947**	**0.739**	**0.831**	**0.817**	**0.889**	-	-	-	0.03
SiamMask[7]	0.692	0.848	0.639	0.743	0.522	0.597	0.559	0.645	0.321	0.686	0.569	43
D3S [124]	0.754	-	0.726	-	0.578	-	0.638	-	**0.439**	**0.699**	**0.769**	25
SiamRPN-SiamSeg	0.663	0.782	0.629	0.707	0.506	0.578	0.552	0.628	0.369	**0.701**	0.651	21
ECO-SiamSeg	0.623	0.744	0.590	0.692	0.503	0.567	0.553	0.620	**0.414**	0.694	0.729	12
ATOM-SiamSeg	0.658	0.785	0.628	0.697	0.544	0.626	0.584	0.664	0.406	0.691	0.723	13
ATOM-SemSeg	0.723	0.846	0.739	0.856	0.566	0.659	0.640	0.751	0.402	0.678	0.735	8
DiMP-SemSeg	0.723	0.827	0.744	0.821	0.583	0.671	0.663	0.765	0.410	0.675	**0.744**	9
DiMP-SiamSeg	0.704	0.844	0.658	0.754	0.553	0.639	0.591	0.675	0.410	0.691	0.730	15

Table 7.8: Comparison of the three experimented segmentation methods with the competitor AlphaRefine method [255] in the baseline experiment on VOT2020. Best segmentation method results, per tracker, are highlighted in red.

Tracker	SemSeg			SiamSeg			FewShotSeg			AlphaRefine [255]		
	EAO$_\uparrow$	A$_\uparrow$	R$_\uparrow$	EAO$_\uparrow$	A$_\uparrow$	R$_\uparrow$	EAO$_\uparrow$	A$_\uparrow$	R$_\uparrow$	EAO$_\uparrow$	A$_\uparrow$	R$_\uparrow$
DCFNet	0.203	0.616	0.426	**0.310**	0.676	0.558	0.230	0.491	**0.567**	0.289	**0.705**	0.511
KCF	0.199	0.648	0.371	**0.285**	0.659	**0.501**	0.200	0.459	0.499	0.269	**0.703**	0.449
SiamFC	0.218	0.602	0.446	**0.309**	0.682	**0.571**	0.228	0.491	0.563	0.305	**0.719**	0.529
MetaCrest	0.240	0.602	0.513	**0.336**	0.657	0.624	0.250	0.479	**0.647**	0.333	**0.712**	0.588
SiamRPN	0.356	0.692	0.639	0.369	0.701	0.651	0.311	0.551	**0.677**	**0.385**	**0.742**	0.637
MDNet	0.295	0.638	0.609	0.371	0.662	0.689	0.308	0.546	**0.723**	**0.382**	**0.720**	0.697
ATOM	0.402	0.678	**0.735**	0.406	0.691	0.723	0.337	0.560	0.731	**0.410**	**0.717**	0.726
DiMP	0.410	0.675	0.744	0.410	0.691	0.730	0.347	0.556	**0.749**	**0.431**	**0.725**	0.729
ECO	0.322	0.632	0.735	0.414	0.694	0.729	0.349	0.561	**0.759**	**0.425**	**0.720**	0.732

In Table 7.8 we report a performance comparison between the studied SemSeg, SiamSeg, and FewShotSeg methods and the competitor approach AlphaRefine [255] on the VOT2020 benchmark. For fair comparison with our methods, we used AlphaRefine just to produce the segmentation mask of the target object while we did not use its tracker correction step. It can be noticed that AlphaRefine has a stronger performance in A$_\uparrow$ in general, while it suffers in the R$_\uparrow$ measure. In the latter measure, FewShotSeg remains the best. For weaker trackers (e.g. DCFNet, KCF, SiamFC, MetaCrest), the overall performance quantified by EAO$_\uparrow$ tells that SiamSeg is the best method. When applied to stronger trackers (e.g. SiamRPN, ATOM, DiMP, ECO), AlphaRefine achieves the best EAO$_\uparrow$ performance.

7.5 Conclusions

In this chapter, we tried to overcome bounding-box state representations in favour of segmentation masks for visual tracking. Our study presented a framework that transforms bounding-box trackers into segmentation trackers. Such a process is achieved by applying a deep learning-based segmentation methods conditioned on the target object after the bounding-box predictions of a tracker. Three segmentation methods have been studied, namely SemSeg, SiamSeg, and FewShotSeg. Their performance was extensively analyzed under different aspects and on different benchmarks. From the study, we concluded that SemSeg and SiamSeg are the stronger methods in general. The combination of such methods with bounding-box trackers like SiamRPN, ECO, ATOM, and DiMP, enabled the development of segmentation trackers that competed with the most recent solutions SiamMask and D3S on the DAVIS 2016 and 2017, and VOT2020 benchmarks.

8 Precise Tracking of the Knee Cartilage in Ultrasound Videos

Knee arthroscopy is a well-established minimally invasive procedure (MIP) for diagnosis and treatment of disorders in knee joints. Its execution requires an initial small incision of the skin and soft tissues of the patient, and the successive insertion of the arthroscope, a flexible scope carrying a small camera, inside the joint. Through a video monitor, 2D images acquired by the camera are displayed to the surgeon, who is able to visualize the anatomical structures of the knee and to guide surgical instruments. Despite being a common procedure nowadays, this kind of intervention demands a great physical and mental effort from surgeons, with the consequent increased chance of damaging the knee structures [263]. To overcome these problems, ultrasound (US) guided knee arthroscopy is currently being studied [264]. US imaging offers accurate and precise anatomical analysis, superior resolution and relative cost-effectiveness. Currently, it is the only real-time volumetric imaging modality that is clinically available and compatible with surgical conditions. The knee is a particularly interesting region amenable to the use of US scanning in surgery-guided applications [265], where most hard and soft tissue structures can be properly identified, segmented and tracked.

The automatic interpretation of 2D+time/3D+time US images of the knee could be a valuable tool able to offer accurate localization and visualization of the knee structures, ultimately reducing surgeon's operating stress. Furthermore, clinicians indicate that knee arthroscopy will be among the first types of MIPs that, in the near future, will be fully automated by robotic surgery [264]. In these scenarios, the automatic interpretation of US images is required [266]. A tracking tool can exploit the visual and temporal information acquired during the intervention, to interpret the variations in position and shape of the knee structures. Such a system would require a minimal user initialization, e.g. a contour or a segmentation and, in comparison with the surgeon, could produce a more accurate and repeatable localization.

Among the structures that are at risk during knee arthroscopy, cartilages are particularly vulnerable [263]. In US images, cartilages are typically clearly visible, but it is

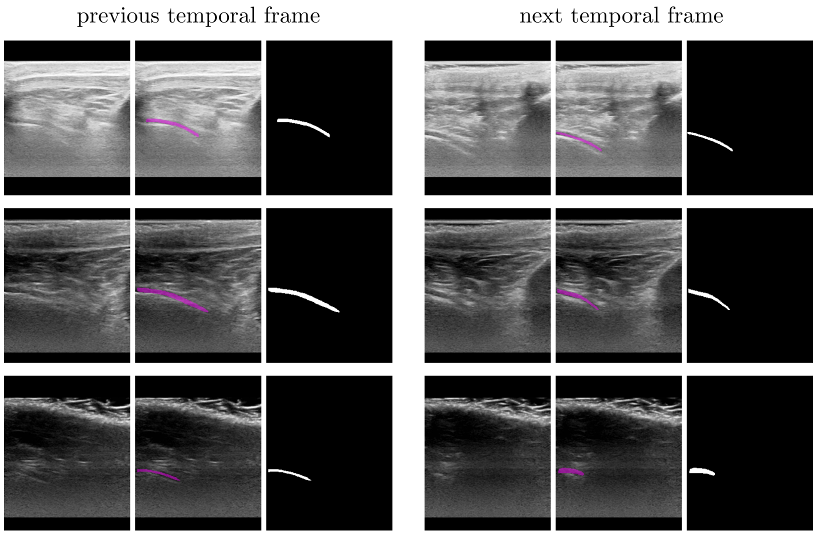


Figure 8.1: Visual examples of US images of the knee, with the highlight of the femoral condyle cartilage. Each of three-image blocks shows a 2D US image, the same US image with the cartilage's ground-truth segmentation (in pink) drawn by a surgeon, and the corresponding binary cartilage mask, respectively. Each row of images shows the transformation of the cartilage from a previous temporal frame of a US sequence to the successive temporal frame. First two rows depict examples of translation of the US probe. The third row presents an example of transformation while the knee is flexing.

not straightforward to track them under surgical conditions, where their position, shape and appearance change due to the physics of the US beam, US probe shifts or knee joint flexion to different angles. In Figure 8.1, US images with the cartilages highlighted are shown.

In the past, several methodologies have been proposed to track anatomical structures in US images, such as tongue [267, 268], heart's left ventricle [269, 270], vessels [271] and liver landmarks [272, 273]. These methodologies included, for example, active contour models and their variations [267, 268], statistical approaches like Kalman filters [271], sparse representation and dictionary learning [270]. One of the biggest limitations of the aforementioned methodologies is that these methods are model-centred and make many assumptions about the problem that may not be realistic. In addition, they also require the development of typically sub-optimal hand-designed representations. To address those issues, deep learning (DL) [274] solutions have been introduced to the field of anatomical structure tracking. DL is a method that automatically learns optimal data representations. For example, [269] combined deep belief networks with a probabilistic non-Gaussian model to track the motion of the left ventricle. [275] proposed convolutional neural networks (CNNs) with a learned distance metric, while [273] developed a deep siamese neural network (SNN).

The latter solution is based on the successful framework of siamese trackers introduced in Chapter 1. But despite the outstanding results achieved on benchmark datasets of natural images, SNN-based visual trackers fail to be applied directly to medical domains due to their high architectural complexity and the unsuitable target object's state representation as bounding boxes.

Building up on the knowledge provided in the previous chapter, in the final chapter of this Thesis we present a DL methodology applied to US images to track the knee cartilage under several clinical conditions during MIP. Our solution combines deep neural networks (DNNs) for the segmentation of medical data and the recent SNN-based framework for visual tracking to track precisely the position and shape of the femoral condyle cartilage with segmentation masks. In particular, the contribution is threefold:

1. The first real-time tracking algorithm for US images of the femoral condyle cartilage;
2. A novel combination of disparate DL architectures, named Siam-U-Net, which merges U-Net [252] and the siamese framework [43, 276];
3. The first use, in the context of visual tracking, of an end-to-end learning strategy that leverages a training loss generally used for segmentation tasks.

To train and evaluate our model, multiple US scans were taken from knees of six volunteers. Volumetric US images were acquired during leg flexion to mimic possible positions of the leg during the intervention, and while the US probe shifted on the surface of the knee. From the US images obtained, given an initial cartilage segmentation, the structure was tracked either in the consecutive US frames, referred as to temporal tracking or both within neighbouring US slices of the same volume and consecutive frames, defined as to spatio-temporal tracking. We show that using segmentation architectures inside the siamese tracking framework is an effective way to localize the femoral cartilage in 2D US sequences with a minimal user intervention. Despite the fact that we propose a 2D+time approach, our solution is fully volumetric, in the sense that it is capable of tracking, both temporally and spatially, the condyle cartilage in any section of 3D+time US sequences.

The proposed solution exhibits a segmentation accuracy, in terms of Dice Similarity Coefficient (DSC) [277, 278], that is comparable to the one produced by two expert operators and that is higher than the segmentation models proposed by [252] and by [279]. Our solution also offers better performance than the state-of-the-art trackers OSVOS [242] and RGMP [243] which were developed for video object segmentation.

8.1 Related Work

Our solution can be placed at the intersection of three research areas: visual tracking, US tracking and medical image segmentation. In this section, we review the most relevant works to our methodology.

8.1.1 Visual Tracking

In its simplest form, the visual tracking problem consists of the consistent recognition of a target in consecutive video frames. The most used target representation is a bounding box that encloses the object of interest. If a more precise localization is needed, a segmentation that identifies the object pixel-by-pixel should be used. In the computer vision literature, the first approach is known as visual object tracking (VOT), while the second is referred as to video object segmentation (VOS).

Visual Object Tracking. In VOT problems, a moving target object must be identified within a searching area (usually bigger than the target) in each video frame. The target is localized in the searching area's sub-region that has the highest visual similarity with the target in the previous frames. In the past years, SNNs have been used for VOT mostly because of their computational efficiency and good accuracy on existing benchmark datasets [41, 43, 280, 85, 93, 230, 44, 45, 47]. An SNN [48] is a particular neural network architecture commonly used to learn representations of two input objects by optimizing a training loss that compares their similarity in higher-level feature spaces. In VOT, this idea is exploited to define a similarity map obtained by comparing the target representation and every sub-matrix of the searching area representation, using as comparison metric the cross-correlation. This solution, known in literature as SiamFC, was firstly proposed by [43]. Subsequently, SiameseRPN [44] increased the detection accuracy by fusing a Region Proposal Network [235] and the cross-correlation operation. [45] proposed to aggregate the CNN features through layer-wise and depth-wise convolutions to enhance the cross-correlation. [47] suggested a siamese architecture to unify the VOT and VOS tasks. Their proposed network is initialized with a ground-truth bounding box and is able to propagate both the box and the segmentation mask that identify and localize the target object through the video.

All these methods have high performance in terms of speed, as they are able to produce the target representations in real-time, i.e. they are able to process more than 30 images per second. This is clearly an advantage which we want to include in our solution. However, these methods are not directly suited for our problem, because a bounding box representation of the target is not sufficient to produce precise information about the location and shape of the cartilage. Additionally, the CNN employed by [47] has many learnable parameters that are not needed for the problem of tracking a single object like the cartilage and that would lead to overfitting, given the limited number of training examples available for our task. Very deep neural networks can achieve outstanding results, but the main drawback is the necessity of large sets of information rich data. Compared to natural images (on which the presented methods perform well), US images are less informative and thus, networks with less parameters can be used. Lowering the number of parameters reduces the chances of overfitting and increases the processing speed of the network.

Video Object Segmentation. To tackle the VOS problem, different methodologies have been proposed. MaskTrack [281] introduced a pixel-labeling CNN that frame-by-frame refines, through a combination of offline and online learning strategies, the previously detected segmentations. Several other papers [282, 283, 284] used spatio-temporal graph representations to distribute the labels estimates to the pixels of consecutive frames. Alternative approaches independently segmented every single frame [242, 285, 286] using an online training scheme. One of the most relevant works in this direction [242] proposed to use one-shot learning to fine-tune online a Fully Convolutional Network (FCN) [287] which was pre-trained to distinguish target object pixels from the ones of the searching area. This solution allowed to reach superior results, but with the drawback of an online pre-processing time of up to 10 minutes. The employment of SNNs in VOS was firstly introduced by [243], who proposed an encoder-decoder fully convolutional siamese architecture with a global convolution operator that was trained to produce a segmentation mask for every frame, given as input: the current frame, the mask produced at the previous time step and the initial ground-truth mask. Our proposed Siam-U-Net follows a similar approach, but it substitutes the global convolution operation with the depth-wise cross-correlation. This allows to produce a high level activation map, which is then refined by the decoder into a fine-grained segmentation.

Despite the promising segmentation accuracies achieved by the methods described above, their high complexity will not allow the production of segmentations in a very short time. In fact, these solutions can process from less than an image to a maximum of 10 images per second. Thus, they are not suited for real-time applications like our problem of interest. Moreover, no methodology took advantage of the DSC as a training loss, which was shown to lead to better segmenting performance [288].

8.1.2 Tracking in US Images

Visual tracking in US images has received increased interest in the past. [267] and [268] used variations of active contours to track the motion of the tongue. These methods rely on image gradient and energy-based functions to draw a contour around the edges of the target object. Even though it is a common technique in computer vision, this kind of methods suffer from initialization robustness, which can lead to drifting over time. [271] proposed a real-time algorithm for vessel segmentation and tracking. Their solution used an elliptical model to segment vessels and Kalman filters to track their shape through temporal sequences. A main drawback of this solution is the assumption that anatomical structures can always be represented through elliptical models, thus reducing the generalization capabilities to structures with other shapes. [270] presented a method that employs multiscale sparse representation and dictionary learning to track the endocardial and epicardial contours of the left ventricle. Despite achieving great results, the biggest limitation of dictionary learning is the assumption that samples can be represented by a linear combination of dictionary items. In contrast, our methodology uses convolutional neural networks (CNNs) to build powerful image representations through non linear operations.

Overall, the biggest limitations of the methods above are that they are model-centred or use linear data-driven methodologies. Furthermore, they make assumptions about the problem that may not hold in practice and they sometimes require the development of sub-optimal hand-designed representations.

More recently, DL based methodologies have been applied to US data. [269] fused deep belief networks and multiple dynamic models by means of a probabilistic non-Gaussian state-space distribution to track the left ventricle. Despite the good results, this method is difficult to be extended to other medical context since the transition model involved takes into account information that is too specific for the cardiac cycle (e.g., it only considers the two cardiac phases of the cycle: diastole and systole). Additionally, the observation model is based on shallow artificial neural networks. In contrast, we employ a CNN based architecture which is proven to work better for spatial data, such as images [20, 18]. [275] proposed a CNN to track liver landmarks in 2D+time US sequences. Their proposed model was trained by optimizing a distance metric between two US image patches. At test time, different image patches were sampled in the current frame around the previous known target location, and the coordinates of the patch with the predicted lower metric value were chosen as new position for the target. We propose a method with a single forward pass, different from the candidate generation procedure proposed by the authors that can harm the processing speed of the tracker, since many image comparisons are to be executed. [273] tackled the liver landmark tracking problem with a SNN and a location prior. This was the first attempt to apply SNNs to US images, but its tracking capabilities are limited to the prediction of the position of the target object, which is represented by the coordinates of a single point. This is not sufficient for our problem of interest that requires precise localization and shape definition of a structure that is characterized by a highly variable appearance.

In general, despite the good reported results, all the approaches mentioned above are not directly applicable to our task because they propose ad-hoc implementations that are optimized for their problem of interest, thus reducing their capability of generalization to other use cases.

8.1.3 Medical Image Segmentation

FCNs for semantic segmentation were firstly introduced by [287]. Their idea was to exploit the knowledge of a CNN pre-trained for natural image classification to perform image segmentation. To this end, the authors added an expanding block to the pre-trained CNN. The block was used to generate the output segmentation by enlarging the CNN intermediate features through convolutional and up-sampling layers. The weights of the newly added module were then learned by means of a supervised segmentation task. This solution showed very good results with respect to previous methodologies [289, 290, 291, 292]. However, the required classification pre-training on the ImageNet dataset [19] is (still today) very computationally expensive and only suited for natural image processing applications. To overcome these problems, [252] proposed a novel fully convolutional architecture, named U-Net, that could be trained end-to-end and with few training samples. The structure of the U-Net extended the one from FCN by [287] and it was the combination of a contracting part (the encoder), composed of convolutional and max-pooling layers, and an expanding part (the decoder), consisting of the aggregation of the encoder intermediate features, up-sampling and convolutional layers. Thanks to its outstanding results in many clinical domains [288, 293, 294, 295, 296], today this methodology is considered the standard architecture for medical image segmentation. Despite this, U-Net has not been effectively adapted to include temporal data. Therefore, U-Net was chosen to form just the base CNN architecture of the cartilage tracker proposed in this chapter. [279] tried to include previously computed segmentation masks into U-Net's architecture as an additional input channel. The idea was to use prior information for aiding the task of 3D segmentation by means of a 2D model. Experimental validation showed the proposed model to be stronger than U-Net in segmenting 3D CT scans of the bladder. In principle, the presented methodology could be applied to track anatomical structures in temporal sequences of 2D images. However, tracking requires fast elaboration times and processing searching areas as large as the image size is usually very time consuming. Moreover, the target object has usual motion patterns that can be exploited to reduce computational time and effort in its

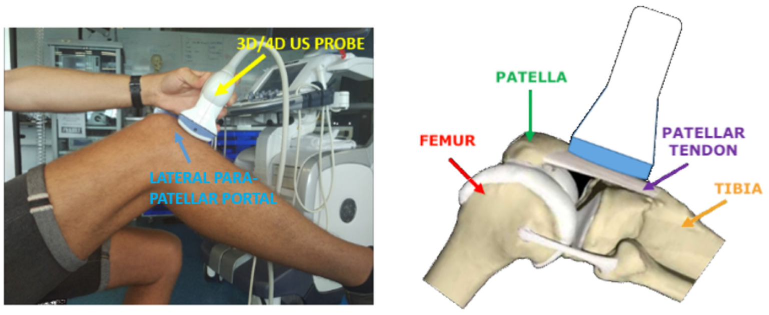


Figure 8.2: US probe positioning. On the left: lateral view of the knee joint with the probe placed on the patellar tendon. On the right: schematic US probe positioning representation, showing the positions of reference structures relative to the probe.

search. The solution proposed by [279] does not take into account these considerations.

8.2 Materials and Problem Formulation

For this study, a dataset of 3D+time images was built by mimicking possible MIP scenarios. In this section we describe how the US data was acquired, labeled and organized. We also give a precise formulation of the problem of tracking the femoral condyle cartilage.

8.2.1 US Data Acquisition and Labels Generation

To build the US dataset, knees of six healthy volunteers (male and female) have been scanned at the Queensland University of Technology using a Philips EPIQ7 US workstation with a VL13-5 mechanically swept probe (Philips Healthcare, Eindhoven, Netherlands). The ethics approval for data acquisition was granted by Queensland University of Technology Ethics Committee (No. 1700001110). All the volunteers signed an informed consent before the data collection.

The US probe was positioned anteriorly to the knee, and the scans were performed through the volunteer's patellar tendons as shown in Figure 8.2. The rationale for this choice was to allow enough space for the insertion and manipulation of the surgical instruments through the medial and lateral parapatellar portals (the soft spots at both sides of the patella), as in realistic intra-operative knee arthroscopy scenarios. The US probe was hand-held by an experienced orthopedic surgeon. The US scans were performed with the knees fully submerged in water to minimize possible acoustic coupling issues. To mimic normal conditions during surgical procedures, we acquired 35 3D+time sequences (3D volumes in time), for a total of 151 full 3D volumes, flexing the knee from 0 to 30 degrees (F30), and translating the probe along the patellar tendon with the knee flexed at 0 degrees (T0) or at 30 degrees (T30). Table 1 reports a summary of the dataset collected. MRI scans of the knees of the same volunteers have also been ac-

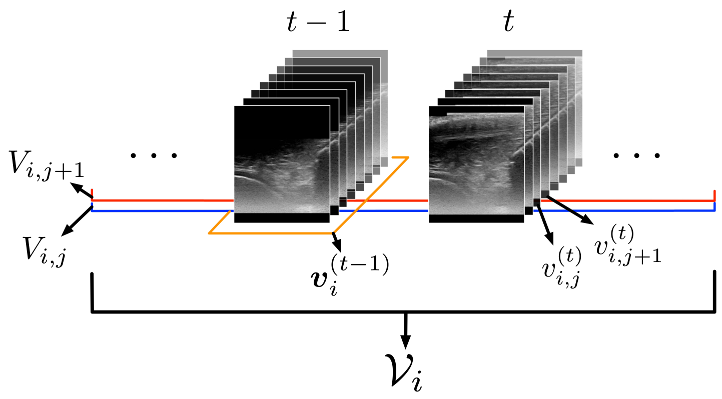


Figure 8.3: Visual representation of the notation used throughout the chapter. Each 3D+time US sequence is denoted as $\mathcal{V}_i$. The volumes belonging to $\mathcal{V}_i$ are referred as $\boldsymbol{v}_i^{(t)}$ (highlighted by the orange line) for the temporal step t. Each 2D+time sequence $V_{i,j}$ (highlighted by the red and blue lines) comprises the slices $v_{i,j}^{(t)}$ which in turn belong to the volumes $\boldsymbol{v}_i^{(t)}$ respectively.

quired in identical geometric conditions and manually fused with the US volumes by an experienced surgeon to accurately identify all the anatomic structures. During knee flexion, the expert operator always tried to capture the US volume from the lower end of the patella to the upper end of the tibia longitudinally, and containing the articular cartilage on both sides of femoral condyles transversely. The US volumes collected had a size of approximately $(4 \times 4 \times 3)$ cm^3 and were acquired at 1 Hz refresh rate.

In the images, typically the femoral cartilages appearance is an hypoechoic band on top of a clear hyperechoic line outlining the bone contour of the femoral condyles. The border between the cartilage layer and Hoffa's fat pad is also typically clearly visible as a thin hyperechoic line parallel to the bone contour. The pixel dimensions are $\sim$0.19mm. The reference segmentations of the femoral cartilages have been manually created by an expert orthopaedic surgeon (Operator 1), along the sagittal slices within the US volumes acquired using MeVisLab (MeVis Medical Solutions AG, Germany). The total number of annotated slice was 18278.

8.2.2 Problem Formulation

The resulting dataset used for this study is composed of a set of temporal sequences of 3D+time US images and respective labels. We denote it as $\mathcal{D}_{3D+time} = \left\{\left(\mathcal{V}_i, \mathcal{G}_i\right)\right\}_{i=1}^{35}$, where each pair $\left(\mathcal{V}_i, \mathcal{G}_i\right)$ is obtained from ordered sequences of volumes $\mathcal{V}_i = \left\{\boldsymbol{v}_i^{(t)}\right\}$ and $\mathcal{G}_i = \{\boldsymbol{g}_i^{(t)}\}$, $t \in \{0, \dots, T-1\}$, $T \in \mathbb{N}$. Each $\boldsymbol{v}_i^{(t)} \in \{0, \dots, 255\}^{r \times c \times d}$ is a US volume of $r \times c \times d$ voxels (in our case $r = 313, c = 255, d = 256$) and $\boldsymbol{g}_i^{(t)} \in \{0, 1\}^{r \times c \times d}$ is the respective reference segmentation volume. Each 2D+time sequence $V_{i,j}$ is composed by considering each $v_{i,j}^{(t)} \in \{0, \dots, 255\}^{r \times c \times 1} \subset \boldsymbol{v}_i^{(t)}$, $j \in \{0, \dots, d-1\}$, i.e. the 2D matrix component (belonging to the volume $\boldsymbol{v}_i^{(t)}$) which we refer as slice, for which the 2D

Table 8.1: Summary of the dataset collected for the study. For each volunteer, we report the scanned legs (L: left, or R: right), the scan type (probe translation with the knee at 0 (T0) or 30 degrees (T30) flexion, or knee flexion from 0 to 30 degrees (F30), the number of volumes acquired and the number of 2D US slices contoured by the expert Operator 1.

Subject id	Leg scanned	Scan modalities	# volumes	# annotated slices
1	L, R	T0, T30, F30	28	3402
2	L, R	T0, T30, F30	24	3245
3	L, R	T0, T30, F30	29	2657
4	L, R	T0, T30, F30	28	3119
5	L, R	T0, T30, F30	23	3872
6	L, R	T0, T30, F30	19	1983

mask $g_{i,j}^{(t)} \in \{0,1\}^{r \times c \times 1} \subset \boldsymbol{g}_i^{(t)}$ presents a localization of the cartilage. In formal terms $V_{i,j} = \{v_{i,j}^{(t)} \mid \forall t \; \exists g_{i,j}^{(t)} \neq 0^{r \times c \times 1}\}$. In Figure 8.3, we show a visual representation of the notation employed in this chapter.

The entire dataset is divided into training and testing sets subject-wise, i.e. with no overlap in terms of volunteers in the training and testing sets. In Table 8.1, details about the acquired data are reported, while in Figure 8.4, the distribution of the contoured slices is shown for each subject.

The use of sequences of 2D data, and so following a 2D+time tracking approach (instead of a 3D+time approach), was motivated by the fact that this setting allowed significantly less computational effort for data processing. In fact, dealing with sequences of 3D volumes would have required the reduction of the volumetric dimensions of the data to fit in the memory of currently available machines, with the consequent loss of valuable information. A 3D+time approach would also need a much larger amount of labeling effort to produce sufficient samples for making DL methods work well, given that each volume can have up to 203 2D annotated slices. Moreover, some of 3D volumes acquired in this study were just partially contoured (i.e. not all the 2D slices composing the volumes and effectively containing a cartilage were segmented by the expert) making them unusable for 3D+time processing.

Our problem of interest is the precise localization of the femoral condyle cartilage in each of the 2D slices that compose a 2D+time US sequence, given an initial 2D reference segmentation for the first slice of the sequence. In formal terms, given a temporal sequence $V_{i,j}$, containing T slices and an initial reference segmentation of the cartilage $g_{i,j}^{(0)}$, drawn by an expert, our method will produce the masks $s_{i,j}^{(t)} \in \{0,1\}^{r \times c \times 1}, t \in \{1, \ldots, T-1\}$ that successfully locate the femoral cartilage. With this setting, the cartilage location and shape representations, $s_{i,j}^{(t)}$, are expressed as binary segmentations.

8.3 Method

The key idea of this study is to combine an encoder-decoder neural network architecture such as U-Net [252] with the siamese tracking framework [43, 276]. We begin this section by describing the novel DL architecture, Siam-U-Net, that is used to produce a cartilage segmentation within a 2D US image, given the information about the structure's visual appearance in the previous time frame and the searching area where the cartilage is

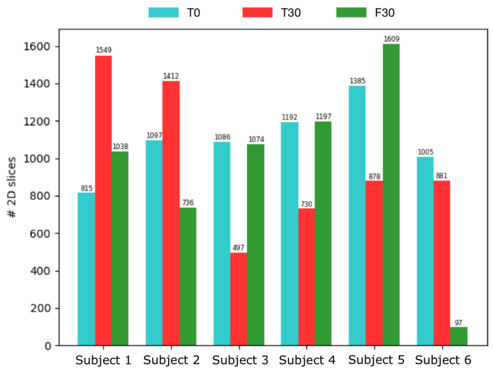


Figure 8.4: The distribution of the 2D contoured slices shown for each subject included in the dataset.

supposed to be present. After discussing training procedure of the network, we introduce how the architecture is used to effectively track the cartilage in a 3D sequence.

8.3.1 Siam-U-Net Architecture

The neural network architecture we propose takes inspiration from the encoder-decoder architecture of U-Net [252], and the cross-correlation operation used in the traditional siamese framework for visual tracking. A graphical representation of the proposed network is depicted in Figure 8.5.

The network receives as input two cropped images, a smaller one for the target cartilage and a bigger one for the searching area. These image crops are passed through the encoder branch denoted as $E_{\theta_E}(\cdot)$, whose weights θ_E remain the same for the two inputs. The encoder is composed of a sequence of five computational blocks each including a set of 3×3 convolutional layers and 2×2 max pooling operators applied with a stride of 2 to reduce the size of the feature maps. Each convolutional layer is followed by batch normalization [297], ReLU activation and a dropout [298] layer.

After the target and searching area are processed by the encoder, the cross-correlation operation is performed. The target representation is depth-wise, i.e. feature map by feature map, cross correlated to the searching area representation, as proposed by [276, 45]. This procedure is implemented as a convolutional layer applied to the searching area feature maps, using the target embedding as convolutional kernel. Zero-padding is applied to the cross-correlated feature maps to match the dimensions of the searching area embedding. The depth-wise cross correlation allows the comparison of the target cartilage image with the slice area where it is supposed to be present. The output of this operation encodes implicit information about the position of the cartilage inside the searching area into a three-dimensional representation, and is indeed a similarity map that is richer than the bi-dimensional one produced by the standard cross-correlation operation. Moreover, to make the correlation meaningful, the weights θ_E of the encoder are shared for the two input images. Since these belong to the same image domain, it makes sense to learn the same hierarchy of features and so to apply

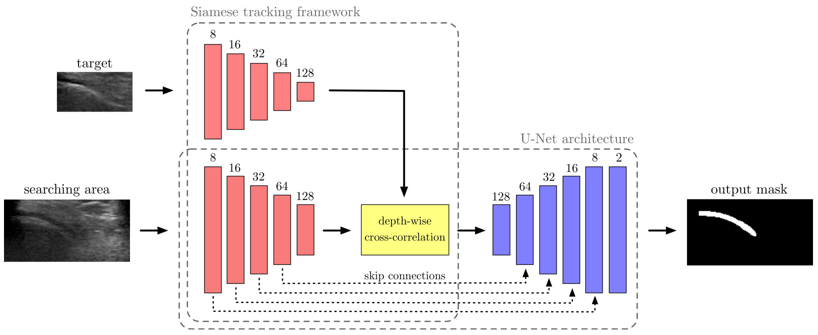


Figure 8.5: Graphical visualization of the novel DL architecture, Siam-U-Net, proposed to track the femoral condyle cartilage. The network takes as input the target and the searching area (showed on the left) which are passed through the encoder $E_{\theta_E}(\cdot)$ represented by the red blocks. Then the target representation is depth-wise cross-correlated to the searching area representation. This operation encodes the information regarding the relative position of the cartilage inside the searching area. This embedding is combined with the intermediate feature maps produced by the encoder on the searching area (skip connections), and it is used by the decoder $D_{\theta_D}(\cdot)$ (blue blocks) to build the segmentation of the cartilage inside the searching area. The values above each block indicate the depth of the feature maps. The rectangles with dashed borders enclose the siamese tracking framework and the U-Net architecture that were used to create this novel network.

the same transformation to the two patches.

After the cross-correlation, the output binary mask is built by the decoder branch $D_{\theta_D}(\cdot)$ that uses four blocks composed sequentially of: the bilinear up-sampling of the feature maps of the previous layer, followed by a 2×2 convolution; a concatenation with the feature representations produced by each encoder block on the searching area (in the literature referred as to skip-connections); and two 3×3 convolutional layers. The latter are followed by batch normalization, ReLU and dropout. To generate the output segmentation, a 1×1 convolutional layer with two output channels is employed after the last block. The first output channel is for the prediction of the foreground object. i.e. the cartilage, while the second one is for the prediction of the pixels belonging to the background of the slice. This last layer is followed by a softmax activation function. The idea here is to refine the high level similarity map produced by the depth-wise cross-correlation operation through the layers of the decoder. Skip connections coming from the searching area branch are used to provide lower level (hence, more detailed) feature context and consequently compute a more fine-grained segmentation of the cartilage in the searching area.

In contrast to U-Net, which uses blocks with 64, 128, 256, 512, 1024 convolutional feature maps respectively, we implemented lighter blocks (i.e. they are composed of a smaller number of parameters) with 8, 16, 32, 64, 128 convolutional feature maps respectively. This modification was done to reduce the computational effort and improve the processing speed of the network. In addition, we took advantage of the dropout layer to improve generalization.

8.3.2 Training Procedure

We trained Siam-U-Net end-to-end using the US data acquired as described in Section 8.2.1. To compose the training mini-batches, two slices belonging to the same subject, to the same leg, to the same US scanning modality and to the same 3D+time sequence were sampled. The first sampled slice was chosen inside the volume of temporal index $t-1$ at slice index j, i.e. $v_{i,j}^{(t-1)}$, while the second sampled slice was randomly chosen among

$$\left\{ v_{i,k}^{(t-1)}, v_{i,k}^{(t)} \;\middle|\; \begin{array}{l} v_{i,k}^{(t-1)}, v_{i,k}^{(t)} \in V_{i,j}, \\ k \in \{j - S_{max}, \ldots, j-1, j, j+1, \ldots, j + S_{max}\}, \\ S_{max} \in \mathbb{N} \end{array} \right\} \tag{8.1}$$

that is the set of spatially near slices that either belong to the $(t-1)$-th or to the t-th volume. Each mini-batch is composed of B pairs, sampled uniformly from intra-volume and inter-volume slices. We believe that useful information for the temporal tracking can be acquired also intra-volume (e.g. from the cartilage anatomical variations between spatially near slices), as this setting could provide changes of the cartilage appearance that are similar to the ones that could be found in inter-volume tracking. In addition, this process allows to augment the number of training samples, with the potential of improving generalization.

Before being fed to the SNN, both target and searching area were resized to $height \times width \times channels$ (in practice $[48 \times 80 \times 1]$ pixels for the target and $[64 \times 160 \times 1]$ pixels for the searching area) by respecting the aspect ratio of the cartilage. The fixed size for the searching area was obtained by assuring that: 1) the resizing process of the cropped slice would not alter the visual aspect of the cartilage; and 2) the feature maps produced by the encoder $E_{\theta_E}(\cdot)$ would be large enough to contain meaningful information. In a similar way, in order to guarantee that the target representation was informative enough, we used resizing dimensions that satisfied the architectural constraints (imposed by the max-pooling operations that halve the feature maps' dimensions) of the encoder and that allowed the feature maps to keep enough spatial information.

The training objective was set to reduce the DSC dissimilarity [288], referred as to DSC loss, between the masks outputted by the network and the reference segmentation of the second slice of the input pair. This is novel in the panorama of VOS, where the Cross Entropy (CE) loss is often utilized. The use of the DSC loss as training cost is motivated by its robustness against class imbalance.

8.3.3 Tracking Procedure

In this section we describe how the presented network is employed to continuously track the knee cartilage in a 2D+time sequence.

Given a US sequence, two temporal consecutive slices at each time step are considered. For the first one a segmentation estimate is known, while for the second one it must be produced by Siam-U-Net. Given $v_{i,j}^{(t-1)}, v_{i,j}^{(t)} \in V_{i,j}$ as consecutive slices and

$$b^{(t-1)} = [x_{tl}^{(t-1)}, y_{tl}^{(t-1)}, x_{br}^{(t-1)}, y_{br}^{(t-1)}] \tag{8.2}$$

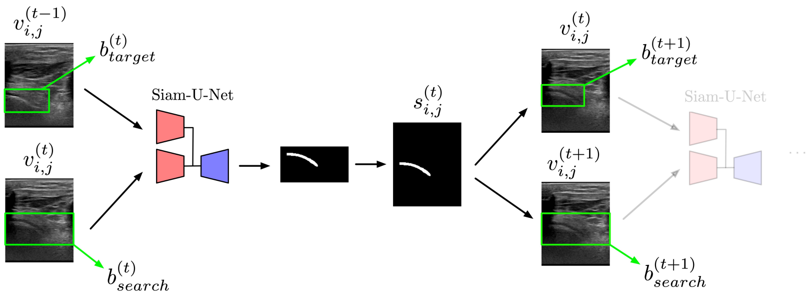


Figure 8.6: Schematic view of the proposed cartilage tracking procedure. On the left, the two consecutive slices $v_{i,j}^{(t-1)}, v_{i,j}^{(t)}$ are cropped by the bounding boxes $b_{target}^{(t)}$ and $b_{search}^{(t)}$ (represented in green), respectively. The two cropped images are fed to Siam-U-Net, which produces the segmentation of the target cartilage inside the searching area. The prediction mask $s_{i,j}^{(t)}$ is then assembled by placing the output mask at the coordinates of $b_{search}^{(t)}$. $s_{i,j}^{(t)}$ is later used to compute $b_{target}^{(t+1)}$ and $b_{search}^{(t+1)}$ in order to crop the slices $v_{i,j}^{(t)}$ and $v_{i,j}^{(t+1)}$

.

the smallest bounding box (defined by the top left and the bottom right vertices) enclosing the non-zero elements of the segmentation at time step $t-1$, $s_{i,j}^{(t-1)}$, the target crop is defined in $v_{i,j}^{(t-1)}$ as follows

$$b_{target}^{(t)} = [x_{tl}^{(t-1)} - P_1, y_{tl}^{(t-1)} - P_1, x_{br}^{(t-1)} + P_1, y_{br}^{(t-1)} + P_1], \tag{8.3}$$

where P_1 is a scalar that allows to enlarge the bounding box in order to include some context area around the cartilage segmentation. The searching area crop is obtained in $v_{i,j}^{(t)}$ as follows

$$b_{search}^{(t)} = [0, y_{tl}^{(t-1)} - P_2, c, y_{br}^{(t-1)} + P_2], \tag{8.4}$$

where P_2 is a scalar used to vertically increase the image context for this slice region. The definition of this crop area is based on two assumptions: 1) the physical layout of the data acquisition strongly limits vertical shifts of the cartilage and 2) the motion of the probe during US acquisition prevents the definition of horizontal shifts limits. Therefore, we selected the whole width of the slice and a limited vertical zone expressed by P_2 as crop area. The two cropped images are fed to the Siam-U-Net which outputs the binary segmentation that locates the cartilage inside the searching area. The output mask $s_{i,j}^{(t)}$ is constructed by placing Siam-U-Net's output mask inside a matrix filled with zeros at the coordinates of $b_{search}^{(t)}$.

At the beginning of the tracking process, the known estimate of the cartilage, $s_{i,j}^{(0)}$, is set to be the reference contour $g_{i,j}^{(0)}$, i.e. $s_{i,j}^{(0)} := g_{i,j}^{(0)}$. In the next step, the segmentation produced by the network, $s_{i,j}^{(1)}$, is used to crop the target and the search area inside the slices $v_{i,j}^{(1)}, v_{i,j}^{(2)}$ respectively. This process is then repeated for all the slices that compose the sequence. The described procedure is depicted in Figure 8.6.

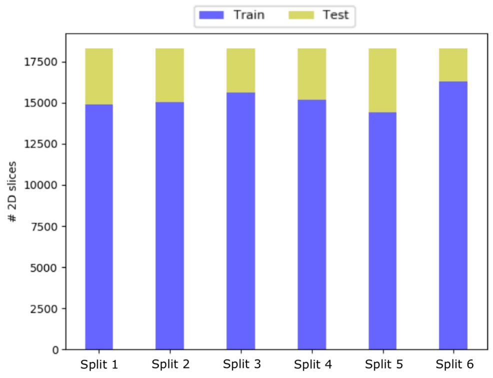


Figure 8.7: Summary of the ratios of training and testing samples in the different experiments done.

8.4 Experimental Setup

In this section we first report how the experimental datasets and procedures have been set up. Then we discuss the error measures employed to validate our methodology. Finally, we present the details of the implementation of the training and tracking procedures.

8.4.1 Dataset Splits

To validate the performance of our solution, we performed a cross validation across the different subjects that compose our US dataset. To this end, we ran six different experiments, where in each one we considered five subjects (80%) for training and one for testing (20%). To optimize the architecture and training hyper-parameters, we ran a first experiment using four subjects for training, one for validation and one for testing. This training, validation and test split was optimized in order to obtain sets with the most similar distributions of samples with respect to the different types of US scans. After their optimization, the hyper-parameters were kept fixed across the six experiments. In Figure 8.7 the distributions of the 2D slice samples considered in the six experiments are shown. Each subject $X \in \{1, .., 6\}$ is used as test subject in the Split X experiment.

8.4.2 Testing Sequences

To evaluate the performance of our methodology we ran Siam-U-Net on all the 2D+time sequences of the subjects who were chosen for testing. In particular, given the sequence $V_{i,j}$ and the initial segmentation $g_{i,j}^{(0)}$ for the slice $v_{i,j}^{(0)}$, we let the tracker run until the end of the sequence, i.e. $\forall v_{i,j}^{(t)} \in V_{i,j}, t > 0$. We then compared each produced prediction mask $s_{i,j}^{(t)}$ with the corresponding reference $g_{i,j}^{(t)}$. In VOT literature this evaluation procedure is referred as to one-pass evaluation (OPE) [71].

Table 8.2: Summary of the test sequences for the temporal tracking setting. Each column reports respectively: the number of test sequences; the total number of slices that have been processed; the average number (± standard deviation) of slices that composed the sequences (i.e. circa 4 slices); the minimum and maximum number of slices in the sequences.

Split	# sequences	# slices	Average sequence length	Min-max sequence lengths
1	849	2224	3.62 ± 1.4	2-6
2	746	1759	3.36 ± 1.1	2-6
3	620	1533	3.47 ± 1.4	2-6
4	720	1701	3.36 ± 1.0	2-5
5	957	2626	3.74 ± 0.8	2-5
6	414	1127	3.72 ± 0.9	2-5

To assess the tracking capabilities of our solution, we set up two testing settings. For the first, we considered all the 2D+time sequences in which each slice belongs to the same volunteer, the same volunteer's leg, the same angle of scanning and the same 3D+time sequence, but to temporally consecutive US volumes. In this way we can assess the temporal tracking capabilities of our solution.

With the second procedure, each pair can include slices belonging either to a consecutive or to the same volume. In the latter case, if $v_{i,j}^{(t)}$ is the first slice of the pair, the second slice is chosen as the nearest slice $v_{i,j\pm1}^{(t)} \in V_{i,j\pm1}$ inside the volume at temporal step t. Given $v_{i,j}^{(t)}$, the pairing slice is randomly selected between $v_{i,j}^{(t+1)}$ and $v_{i,j\pm1}^{(t)}$ using a uniform distribution. We refer this setting as to spatio-temporal tracking.

Tables 8.2 and 8.3 summarize the test sequences used for each split.

8.4.3 Error Measures

For both the temporal and spatio-temporal tracking settings, we measured the DSC [277, 278] between the predictions of Siam-U-Net and their respective reference segmentations. The DSC is a set similarity score that ranges in $[0, 1]$, which is measured as two times the number of overlapping pixels between two binary segmentations, normalized by the sum of the total number of pixels contained in the two. A DSC equal to 0 means that the two segmentations do not overlap, while a DSC of 1 defines a perfect overlap situation. The use of this index was motivated by the fact that it is agnostic to the size of the segmentation. Comparing to a distance-based measure (e.g., Hausdorff distance), DSC enables the computation of results in situations where objects have varying dimensions, which is the case for our problem. Across different slices, the cartilage can be very small (composed of around 4 pixels) or occupy a much larger part of the field of view (up to 1403 pixels). Computing the mean and standard deviation of the Hausdorff distance in this scenario would result in a widespread distribution, hiding the real amount of error made by the model.

As an aggregate metric, we computed the average value (along with standard deviation) of the DSC across all the slices for which a prediction is given by Siam-U-Net. Additionally, the boxplots containing the information regarding the median, the upper and lower quartiles, and range of the DSC values are reported.

Furthermore, we build the success plots for the two testing settings. The success plot [71] is used in VOT to evaluate the accuracy of a tracker and it is built by counting

Table 8.3: Summary of the test sequences for the spatio-temporal tracking setting.

Split	# sequences	# slices	Average sequence length	Min-max sequence lengths
1	849	13633	17.06 ± 11.9	2-69
2	746	9356	13.54 ± 10.3	2-54
3	620	8535	14.77 ± 11.2	2-66
4	720	9808	14.62 ± 10.7	2-54
5	957	14070	15.70 ± 10.5	2-61
6	414	5892	15.23 ± 10.2	2-54

the number of frames that obtained a positive prediction. A prediction is considered positive if the intersection-over-union (IOU) between the predicted and the ground-truth bounding-boxes is above some threshold defined in the range $[0, 1]$, otherwise the prediction is negative. Varying the thresholds for the IOU, different values of accuracy are obtained. With enough samples, [71] showed that the area under the curve (AUC) of the success plot tends to be the average IOU. For our purposes we followed a similar approach, presenting a setup substituting the IOU with the DSC.

8.4.4 Evaluation Procedures

To extensively assess the performance of our methodology we employed six evaluation setups.

Evaluation 1. In this first setup, we evaluated the general performance of our methodology by running Siam-U-Net on all the temporal and spatio-temporal 2D+time sequences obtained from the testing subject's 3D+time sequences. The predicted segmentations were compared with the respective references using the DSC. The distribution of the predictions was assessed by mean, standard deviation, boxplots and success plots. The processing speed of the network was also determined, by measuring the processing time (in milliseconds) to obtain a prediction. The reciprocal of the average measured time was used to express the number of slices-per-second. Finally, qualitative examples of the predictions were obtained. In this setup, the general capabilities of tracking the cartilage, in real-time, were evaluated.

Evaluation 2. To make sure that Siam-U-Net developed a tracking performance which is consistent and robust through time, we evaluated the performance of our solution at different temporal steps. For the temporal tracking setting, we evaluated the distribution of the DSC after every prediction (i.e. when $t = 1, 2, 3, 4, 5$) by measuring mean and standard deviation. In the spatio-temporal setting instead, the same distribution was evaluated after each temporal step (i.e. when two consecutive slices belonged to different volumes), and after $J = 1, 3, 5, 10, 15$ slices processed inside each volume $\boldsymbol{v}_i^{(t)}$. Both results were obtained considering the DSC distributions across all the six experiments.

Evaluation 3. To further establish that Siam-U-Net learned an effective tracking ability through its architectural modules, a quantitative and a qualitative examinations were performed on the siamese encoder $E_{\theta_E}(\cdot)$ and the decoder $D_{\theta_D}(\cdot)$. In the first setting, we measured the mean DSC and standard deviation considering the scenario where the $E_{\theta_E}(\cdot)$'s branch processing the target cartilage is not active. This was done by replacing

$E_{\theta_E}(\cdot)$'s branch intermediate features with a zero filled tensor, before being inputted to the depth-wise cross correlation layer. In this way, we can assess the importance of the information encoded by the target patch branch, and the robustness of the searching area branch in providing meaningful features for producing segmentations without the target cartilage. For the second setup instead, given a target cartilage image, different runs of Siam-U-Net with vertically shifted searching areas were performed. The activations of the decoder's feature maps after block 2, 4 and the output respectively, were visualized as heatmaps by reducing the range of the computed values in [0, 1] (by subtracting the minimum of the values and then dividing by the width of the range). The intention of this test was to examine the decoder's learned features in reflecting effectively the position variations of the target cartilage inside the searching areas.

Evaluation 4. To support the use of the DSC as training loss, a comparison between Siam-U-Net trained with the DSC loss and the same network trained using the CE loss was done. For the CE loss setting, the same architectural and training hyper-parameters used for the DSC loss were maintained. The two different networks were then tested by measuring the average DSC and standard deviation using the temporal test sequences presented in Table 8.2. The predictions of the two obtained models were also evaluated qualitatively.

Evaluation 5. The assessment of Siam-U-Net against the expert performance was based on a comparison with the intra-operator error. Six US volumes (two for every scanning modality) were re-annotated by Operator 1, and a second expert (Operator 2) was asked to contour them. The volumes were randomly chosen by making sure that they would vary among different volunteers, legs and scanning angles. In two separate sessions, each expert was provided with one volume at a time and asked to contour the cartilage on each of the sagittal US slices comprised in that volume. This was done to measure the annotator consistency in outlining the femoral cartilage, avoiding the introduction of other possible sources of variability in the intra-observer study. After that, the DSC between the new and the reference annotations was computed in order to estimate the experts' consistency. The distribution was again evaluated through mean, standard deviation and a boxplot. We also assessed the p-values of a two-sample test [299] to evaluate the correlation between the DSC distributions of: Operator 1 and Operator 2; Siam-U-Net and Operator 1; Siam-U-Net and Operator 2.

Evaluation 6. To further validate our proposed methodology, a comparison with state-of-the-art segmentation models was performed. In particular, we implemented U-Net following the architectural details provided by [252]. U-Net was trained by optimizing the DSC loss with the Adam optimizer [90] for 30 epochs with an initial learning rate of 10^{-4} that was successively halved at epochs 10 and 20. Batches of 24 slices were used. A weight decay of $5 \cdot 10^{-4}$ was also added as regularization term. A comparison with the solution of [279] was also performed. As suggested by the authors, an extra input channel containing a binary mask of the cartilage was added to U-Net's architecture. The proposed model was trained to perform cartilage's contour propagation. Given as inputs a previously known segmentation of the cartilage and a US slice, the network shall predict the segmentation that localizes the cartilage inside the US image. The model was trained with the same hyperparameters used for U-Net except for the number of epochs, that was set to three. During training, for each sample, the input binary mask was selected among the 10 reference segmentations $\{g_{i,j+k}^{(t)}, k \in \{-10, \dots, 0\}\}$ adjacent to slice $v_{i,j}^{(t)}$, as detailed by the authors. At test time, the mask outputted by the network at each step is later used as input segmentation at the successive prediction. In addition to the tests above, we performed a comparison with two VOS state-of-the-art methods. In particular, we implemented the solutions of [242] and of [243], which are referred as to OSVOS and RGMP respectively. The former is currently the best performing solution in the single-object VOS panorama, while the latter is the best in terms of processing speed and it is also the solution most similar to Siam-U-Net, as both use SNNs. Both methodologies publicly provided their source code and we adapted them to the acquired US data. Six experiments were run using 5 subjects for training and one for testing, as done for Siam-U-Net. In each experiment, RGMP was trained for 10 epochs using all the 2D+time US sequences, obtained from the training subjects. The only modifications to OSVOS were the use of the Adam optimizer [90] (instead of the Stochastic Gradient Descent algorithm), the learning rate of 10^{-4} and the number of epochs (500). These were done in order to reduce the online training time (from 10 minutes to circa 3).

For all the experimental setups, after training, the models were then tested with the 2D+time sequences obtained from the testing subject in the temporal tracking setting (which were presented in Table 8.2). As done for Siam-U-Net in Evaluation 1, the average DSC, standard deviation, boxplots and the number of slices-per-second were measured.

8.4.5 Implementation Details

In this section we report the results of the hyperparameters search which led to the best performance on the validation set.

Before being fed to the neural network, the target and searching area were resized to $[48 \times 80 \times 1]$ pixels and $[64 \times 160 \times 1]$ pixels, respectively. In our dataset, the average dimensions of the bounding boxes enclosing the target were 36 pixels in height and 72 pixels in width. The average dimensions for the searching areas were 40 pixels and 160 pixels. The padding values were set to $P_1 = 8$ pixels and $P_2 = 20$ pixels. Successively, the cropped and resized images were normalized by dividing each pixel value by 255. Before the cropping and resizing of the target and the searching area, each slice and its respective reference mask were resized to $[196 \times 160 \times 1]$ pixels to improve the speed of the network while processing smaller images. The dimensions were chosen making sure that the resized slices had an aspect ratio similar to the original slices. Using the validation set, we evaluated that this resizing process caused a performance loss (in terms of DSC) of around 1%, but it allowed an improvement of $\times 1.6$ in the processing speed of our solution.

The model was trained for 75000 iterations using the Adam optimizer [90]. The initial learning rate was set to 10^{-4}, and then halved two times, at iterations 45000 and 60000, respectively. A weight decay of 0.0005 was also added to the DSC loss as regularization term. Each mini-batch was composed of $B = 64$ pairs. In the composition of training pairs, the number of possible nearest slices S_{max}, was set to 10. We experimented removing the constraint of choosing just the S_{max} nearest slices and instead we composed

Table 8.4: Results of Siam-U-Net obtained, on Evaluation 1, for the temporal (left column of results) and for the spatio-temporal (right column) tracking settings.

Split	Temporal tracking average DSC	Spatio-temporal tracking average DSC
1	0.74 ± 0.15	0.73 ± 0.16
2	0.69 ± 0.20	0.71 ± 0.16
3	0.69 ± 0.16	0.70 ± 0.15
4	0.69 ± 0.17	0.68 ± 0.18
5	0.73 ± 0.14	0.73 ± 0.14
6	0.69 ± 0.15	0.68 ± 0.16
Total	0.70 ± 0.16	0.71 ± 0.16

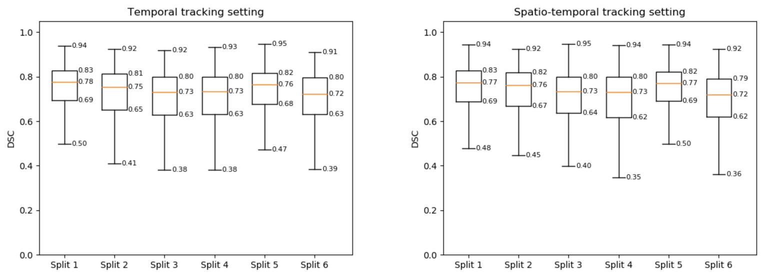


Figure 8.8: Boxplots for Evaluation 1. Each boxplot shows the DSC distribution per experiment. On the left, the plots for the temporal tracking setting are presented. On the right, the same plots but for the spatio-temporal setting.

training pairs of random inter and intra volume slices. The motivation for this was to learn the most generic transformations of the cartilage, however this setup did not achieve good performance. The rate of the Dropout layer was set to 0.4.

At test time, no online update of the network's parameters was performed. Additionally, the foreground output masks $s_{i,j}^{(t)}$, that had a size of $[196 \times 160 \times 1]$ pixels were resized to match the size of the reference segmentations, which is $[313 \times 255 \times 1]$ pixels.

Experiments have been conducted running our Python code with the PyTorch [300] machine learning framework on an Intel Xeon E5-2690 v4 @ 2.60GHz CPU with 320 GB of RAM, four NVIDIA TITAN V GPUs and an NVIDIA TITAN Xp GPU each with 12 GB of memory. The training took around 7 hours.

8.5 Results and Discussion

Evaluation 1. In Table 8.4 and in Figure 8.8, we show the results achieved for Evaluation 1.

The average DSC across all experiments is 0.70 ± 0.16 for the temporal tracking setting while it is 0.71 ± 0.16 for the spatio-temporal setting. The median averaged between the six experiments resulted in 0.75 for both settings. The boxplots show compact distributions of the predictions. The low difference between the results of the two settings suggests that the proposed model is robust to the increased length of the sequences and it is able to overcome the variations of the cartilage appearance both in

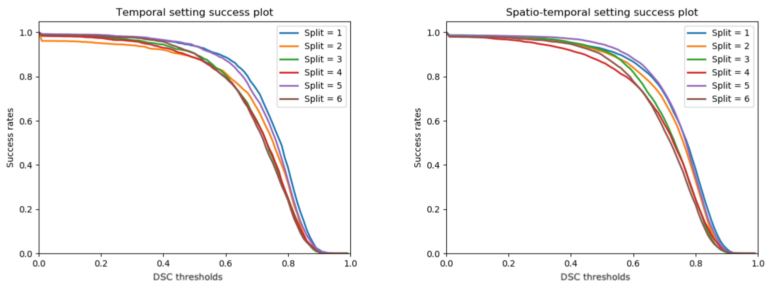


Figure 8.9: Success plots, for Evaluation 1, of the temporal (left image) and of the spatio-temporal tracking settings (right image).

inter and in intra volume scenarios.

The results here obtained do not depend on the dataset split, thus on the subject, the knee and the scan type. This indicates that our solution captures the variability that occurs among different subjects and is able to generalize well to new cases.

The success plots for the temporal and spatio-temporal experimental scenarios are presented in Figure 8.9. It can be seen that Siam-U-Net presents a high percentage ($> 80\%$, on the vertical axis) of predictions that have a DSC with the reference of at least 0.6 (shown on the horizontal axis). When more precise segmentations are considered, i.e. with a DSC > 0.6, the performance of our methodology quickly drops. This is in part explained by the fact that the number of pixels that compose the segmentations of the cartilage is very low with respect to the number of pixels in the slices (as an average computed on the entire dataset, just $\sim$1% of all pixels belong to the cartilage). This causes the DSC to decrease rapidly if just a few pixels are misclassified by the algorithm.

In terms of speed, our solution runs at $\sim$90 slices-per-second on the machine detailed in Section 8.4.5. Since in the computer vision literature, 25-30 frames-per-second are considered real-time performance, we can state that Siam-U-Net is able to run in real-time.

In Figure 8.10 we present some qualitative results of our proposed solution. In the left block of the figure, going from left to right the three images show respectively the US slice $v_{i,j}^{(t-1)}$, $v_{i,j}^{(t-1)}$ with the reference segmentation $g_{i,j}^{(t-1)}$ (in pink), and $v_{i,j}^{(t-1)}$ with Siam-U-Net's prediction $s_{i,j}^{(t-1)}$ (in green) for the temporal step $t-1$. In the right block, each image shows the same elements, but for the next temporal step t. Each row of the figure shows a different US sequence.

Evaluation 2. In Table 8.5, the results of the temporal tracking consistency evaluation are reported. After the first prediction, Siam-U-Net's DSC performance decreases by 4% on average, showing robustness for tracking. This result also shows that the proposed model has a small performance loss when it uses target patches that are not properly aligned with the actual shape and position of the cartilage, i.e. they propagate some error from previous predictions. With this performance, we can say that Siam-U-Net's tracking ability is also robust to target initialization errors.

In Table 8.6 we present the results of the consistency assessment in the spatio-

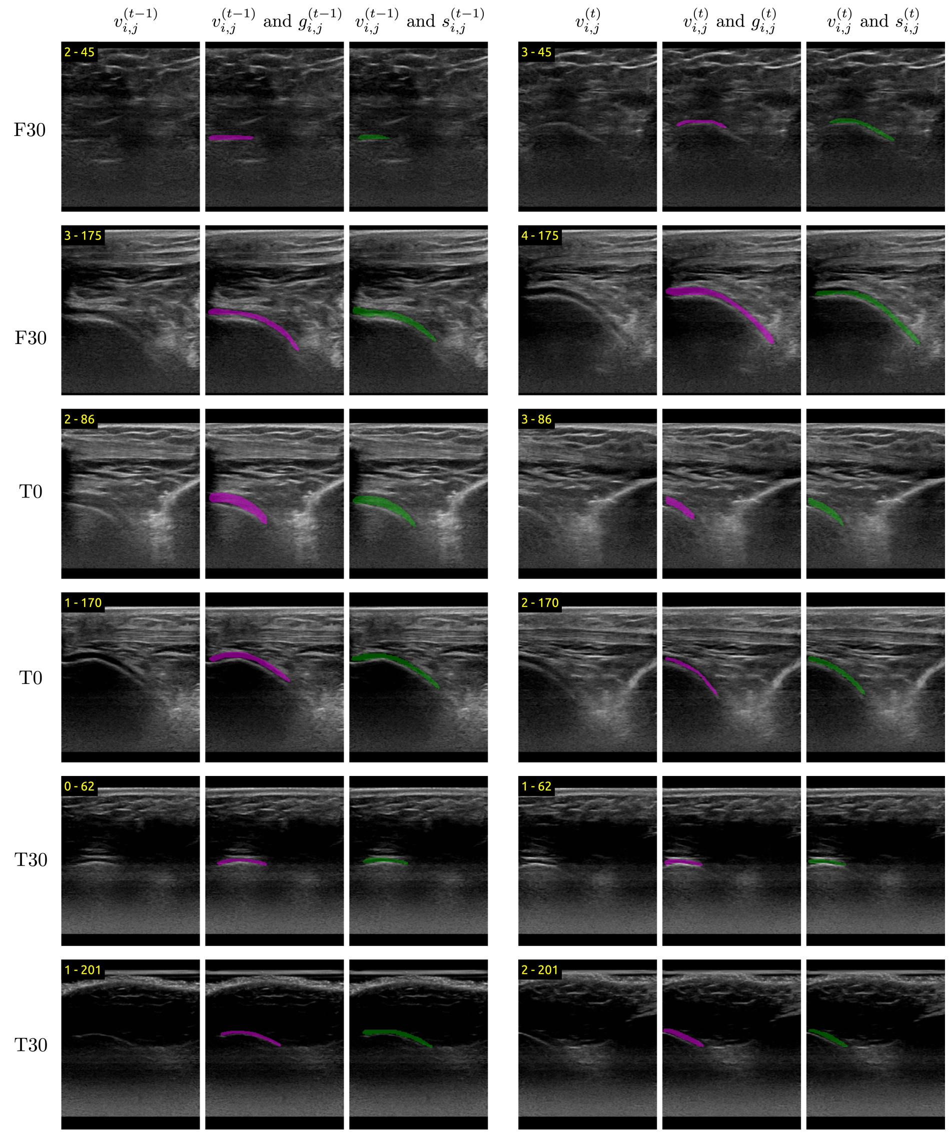

$v_{i,j}^{(t-1)}$
$v_{i,j}^{(t-1)}$ and $g_{i,j}^{(t-1)}$
$v_{i,j}^{(t-1)}$ and $s_{i,j}^{(t-1)}$
$v_{i,j}^{(t)}$
$v_{i,j}^{(t)}$ and $g_{i,j}^{(t)}$
$v_{i,j}^{(t)}$ and $s_{i,j}^{(t)}$
F30
2 - 45
3 - 45
F30
3 - 175
4 - 175
T0
2 - 86
3 - 86
T0
1 - 170
2 - 170
T30
0 - 62
1 - 62
T30
1 - 201
2 - 201

Figure 8.10: Qualitative results of our proposed algorithm. The left block, composed of three images, shows respectively the US slice, the US slice with the reference segmentation (in pink) and the US slice with the prediction of our algorithm (in green) for the step $t-1$. In column on the right, the US slice, the US slice with the reference segmentation and prediction for the successive step t are presented. Each row corresponds to a different test sequence. On the left of each row of images, the knee scan modality is reported. The two yellow numbers indicate, respectively, the temporal index t and the slice index j.

Table 8.5: Results of Evaluation 2. Mean DSC and standard deviation computed at the different temporal steps t in the temporal tracking setting.

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
DSC	0.73 ± 0.13	0.69 ± 0.18	0.70 ± 0.18	0.68 ± 0.18	0.69 ± 0.18

Table 8.6: Results of Evaluation 2. Mean DSC and standard deviation computed at the different temporal volume indexes t and different spatial indexes J in the spatio temporal tracking setting.

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$J = 1$	0.74 ± 0.13	0.71 ± 0.15	0.69 ± 0.19	0.73 ± 0.16	0.70 ± 0.15
$J = 3$	0.71 ± 0.15	0.71 ± 0.15	0.70 ± 0.18	0.73 ± 0.14	0.71 ± 0.15
$J = 5$	0.70 ± 0.16	0.72 ± 0.15	0.71 ± 0.17	0.73 ± 0.15	0.74 ± 0.11
$J = 10$	0.71 ± 0.16	0.72 ± 0.15	0.70 ± 0.17	0.75 ± 0.14	0.73 ± 0.10
$J = 15$	0.72 ± 0.15	0.72 ± 0.15	0.70 ± 0.17	0.74 ± 0.15	0.73 ± 0.08

temporal setting. Apart for $J = 1, 3$, the performance tend to increase after $J = 5, 10, 15$ slices processed inside the same volume. This demonstrates that tracking through space is easier than tracking through time because of less spatial and appearance changes of the cartilage. After the first processed slice, i.e. $J = 1$, Siam-U-Net's performance decreases by 3.25% across the different volumes, which is consistent with the results presented in Table 8.5. The lower temporal performance loss, together with the general increase of the average DSC across spatial predictions, suggest that tracking in space can help to reconstruct better target and searching area patches which in turn can lead to more accurate future predictions.

In general, Siam-U-Net loses some accuracy with the increased length of the sequences, but the results indicate that our proposed network is able to behave well in situations where different kinds of cartilage motion happen. In particular, we can say that Siam-U-Net developed the capability of overcoming both rigid and non-rigid transformations of the cartilage, the former depending on external events such as probe translations, while the latter depending on the changing aspect of the inner anatomical structures while moving the knee. Thus, the proposed solution effectively learned how the cartilage transforms between consecutive slices. This conclusion can be further supported by the performance on the spatio-temporal experimental setting in which Siam-U-Net had to track the cartilage both between temporal consecutive slices (in which the cartilage shape changed due to the events described above) and the spatially nearest slices (the cartilage shape varies within the acquired volumes).

With respect to the latter situation, we believe that our methodology could be also used, as an operator-aided system, to segment US volumes or portions of them. In this scenario, the system could be inputted with just an initial 2D reference segmentation that would be then propagated iteratively to the spatially nearest slices, ultimately producing a volumetric segmentation.

Evaluation 3. Table 8.7 displays the results of the quantitative evaluation with the encoder's target patch branch disabled. The high discrepancy with the results of the complete architecture demonstrates that previous visual information embedded by the encoder on the target patch is necessary to provide a correct segmentation of the car-

Table 8.7: Evaluation 3. Mean DSC and standard deviation results of executing Siam-U-Net with the target image patch branch disabled.

Split	Siam-U-Net	Siam-U-Net without target branch
1	0.74 ± 0.15	0.35 ± 0.31
2	0.69 ± 0.20	0.18 ± 0.23
3	0.69 ± 0.16	0.17 ± 0.26
4	0.69 ± 0.17	0.14 ± 0.24
5	0.73 ± 0.14	0.26 ± 0.27
6	0.69 ± 0.15	0.60 ± 0.26
Total	0.70 ± 0.16	0.28 ± 0.26

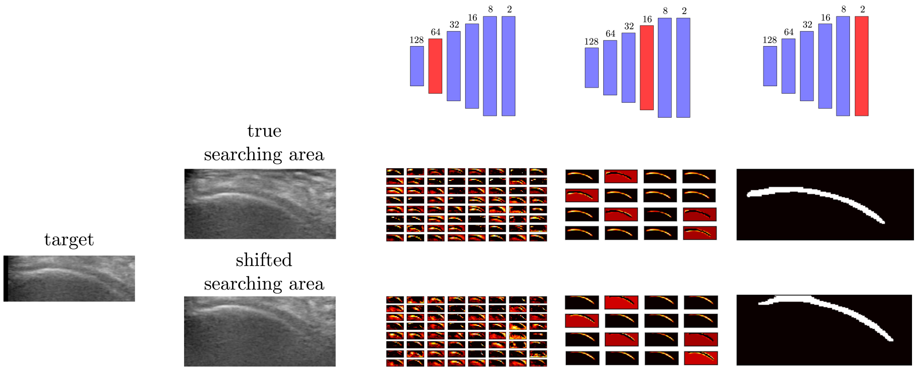


Figure 8.11: Qualitative analysis of the Siam-U-Net's decoder feature activations at different positions of the cartilage. For the same target cartilage slice patch, two vertically shifted searching areas are inputted to Siam-U-Net. The intermediate features of the decoder (which belonging layers are highlighted in red in the first row of pictures) and the output mask reflect the shift happening in the searching area, suggesting that our solution effectively learned to localize the target cartilage.

tilage. This test shows the significance of the temporal information coming from the target patch in the previous slice, with respect to the appearance information of the cartilage included in the current slice.

In Figure 8.11 the qualitative analysis of the Siam-U-Net's decoder feature activations is shown. While maintaining the same target patch, the original searching area (i.e. the one obtained by the bounding box $b^{(t)}_{search}$) and the vertically down shifted searching area are considered. It can be noticed how the activations and the output mask reflect the shift happening in the searching area. This result suggests that the decoder learned to refine the high level localization map produced by the depth-wise cross correlation operation and thus localize effectively the target cartilage in searching areas.

In contrast to classical statistical approaches for tracking where the trade-off between motion and appearance models are in general controllable, in our setting the balance between the two is learned inherently during training. As pointed by [301], SNN-based trackers integrate easily into a single network different tracking-related tasks, such as feature extraction, matching and localization. The proposed Siam-U-Net is an example of that. Although some work has been done from a theoretical point of view [301], we

Table 8.8: Evaluation 4. Comparison of the results obtained in the temporal tracking setting by training Siam-U-Net with the DSC loss and the CE loss respectively.

Split	Siam-U-Net DSC Loss average DSC	Siam-U-Net CE Loss average DSC
1	0.74 ± 0.15	0.61 ± 0.24
2	0.69 ± 0.20	0.65 ± 0.21
3	0.69 ± 0.16	0.69 ± 0.16
4	0.69 ± 0.17	0.68 ± 0.18
5	0.73 ± 0.14	0.67 ± 0.18
6	0.69 ± 0.15	0.68 ± 0.18
Total	0.70 ± 0.16	0.66 ± 0.19

are not aware of papers that have studied the capabilities of SNN module in VOS. An extensive study to analyze in depth how to control the architectural components of SNN for tracking is out of the scope of this study, but by presenting the results of Evaluation 5, we tried to provide a preliminary explanation on the impact of the target branch in the segmentation of the object and of the higher level features that are learned by the decoder.

Evaluation 4. In Table 8.8 we present the comparison between Siam-U-Net trained with the DSC loss and Siam-U-Net trained using the CE loss. The employment of the DSC loss allowed us to produce a more accurate and stable tracking between the different subjects. Through a visual inspection of the resulting segmentations we noticed that the majority of the failure cases of Siam-U-Net trained with the CE loss happened when the hypoechoic and hyperechoic lines of the cartilage were not clearly distinguishable. In these cases, we believe that the CE loss does not produce a learning signal that is meaningful enough for the weak patterns present in these slices. In Figure 8.12 we show some examples of the described situations.

Evaluation 5. The DSC between the reference and the new segmentations annotated by Operator 1 resulted in 0.63 ± 0.30 and median DSC of 0.77. This result was consistent with Operator 2 that had a mean DSC of 0.61 ± 0.25 and median DSC of 0.69. In Figure 8.13 the boxplots for the two observer evaluations are given. It can be easily seen how widespread the two DSC distributions are. The p-value of the two-sample test between the DSC distributions of the experts resulted in 0.242, suggesting a correlation between the two. The comparison between Siam-U-Net's and Operator 1's and Operator 2's performance achieved p-values of $3.41 \cdot 10^{-9}$ and $6.35 \cdot 10^{-15}$ respectively. This shows that there is no correlation between the performance of Siam-U-Net and the one of the experts. Given these results, we can say that Siam-U-Net has an average localization ability that is higher and more robust than the expert operators. The high intra-observer variability can be motivated by the effect of US physics on the knee cartilage, making its localization difficult. Due to US physics, the US beam has a better reflection when it perpendicularly intercepts the part of the cartilage which is flat and consequently it allows to produce an image with better quality in those regions. These situations make easier the distinction of the cartilage hypoechoic and hyperechoic lines. However, it is not the case when the beam intercepts the left and right extremes of the cartilage. Due to the non-perpendicularity of the cartilage walls in those areas, the transmitted US beam are subject to scattering. This leads to images where the cartilage structure is,

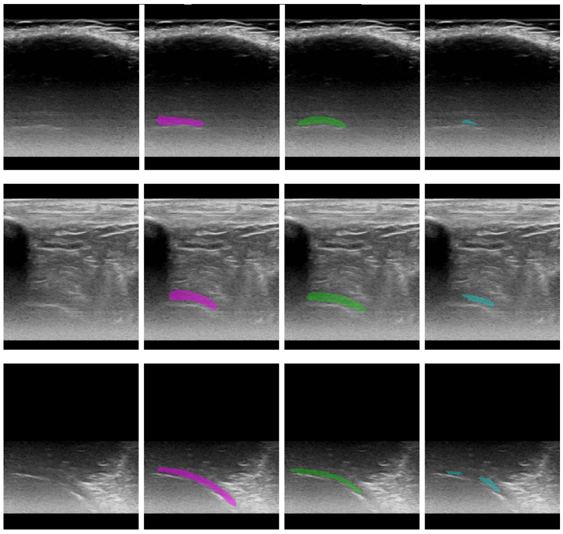


Figure 8.12: Qualitative comparison of Siam-U-Net trained with either the DSC loss or the CE loss. From left to right, the first column of images shows the original US slices; the second the US slices with the reference segmentations; the third the predictions of Siam-U-Net trained with the DSC loss and the last column on the right the predictions of Siam-U-Net trained with the CE loss.

partially or sometimes totally, not visible.

Evaluation 6. U-Net's mean and standard deviation DSC values are reported in Table 8.9 for the temporal tracking scenario while a boxplot is represented on the left plot of Figure 8.14. The average performance is 6% lower than Siam-U-Net, with widespread distributions resembling the expert operators' outcome. This worse performance can be in part explained by the class imbalance of pixel masks. Since U-Net has to predict more pixel probabilities (i.e. prediction masks have bigger dimensions than the ones of Siam-U-Net), it is more susceptible to mislabeling. This situation, together with the small percentage of pixels belonging to the cartilage, makes it easier to missegment the cartilage, increasing the spread of the distribution and decreasing the average performance. Similar conclusions can be reached for the solution by [279]. Regarding the processing time, U-Net predicts segmentations with an average speed of 45 slices-per-second, half the speed of Siam-U-Net, while the solution of [279] runs at 35 slices-per-second. In summary, with respect to a tracking-by-segmentation approach used by the compared works, the use of previous temporal or spatial information and Siam-U-Net's architecture is definitely useful to speed up the tracking process and to provide a more accurate and consistent segmentation of the femoral condyle cartilage.

In Table 8.10 the results of Siam-U-Net against OSVOS and RGMP are reported. We suggest that the lower performance of both OSVOS and RGMP are caused by overfitting, due to the relatively small dataset used and the high capacity of the models, that are composed by very deep CNNs. In terms of processing speed, the test revealed

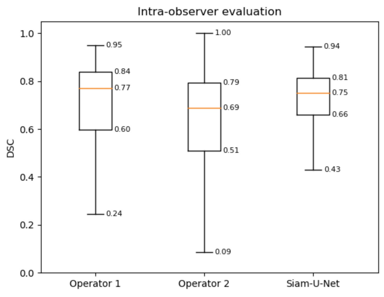


Figure 8.13: Boxplots for the intra-observer evaluation (Evaluation 5) on the two expert operators and for Siam-U-Net. The boxplot for Siam-U-Net was obtained by considering all the predictions across the six dataset splits.

Table 8.9: Results of Evaluation 6. Comparison of Siam-U-Net performance against U-Net [252] and the model proposed by [279].

Split	Siam-U-Net average DSC	U-Net average DSC	[279]'s U-Net average DSC
1	0.74 ± 0.15	0.61 ± 0.24	0.60 ± 0.23
2	0.69 ± 0.20	0.62 ± 0.22	0.65 ± 0.23
3	0.69 ± 0.16	0.68 ± 0.18	0.68 ± 0.21
4	0.69 ± 0.17	0.62 ± 0.23	0.64 ± 0.22
5	0.73 ± 0.14	0.66 ± 0.21	0.67 ± 0.20
6	0.69 ± 0.15	0.63 ± 0.23	0.62 ± 0.28
Total	0.70 ± 0.16	0.64 ± 0.22	0.64 ± 0.23

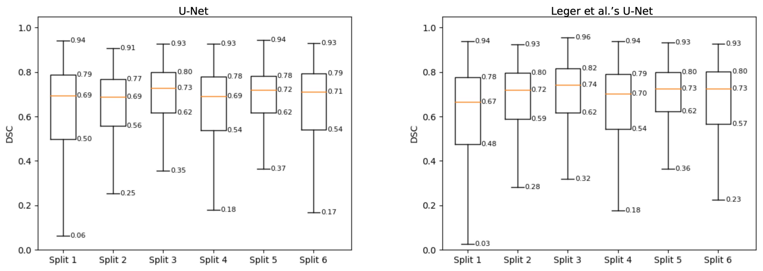


Figure 8.14: Boxplots for the temporal tracking performance of U-Net (on the left) and of the solution of [279] (on the right).

that RGMP had an average running time of around 38 slices-per-second, about two times slower than Siam-U-Net. OSVOS processed around 7 slices-per-second, with an additional time of 3 minutes for the online training that is performed before processing every 2D+time sequence. Siam-U-Net instead is trained solely offline and it can be applied straight away to any given sequence of images. Additionally, the end-to-end strategy employed by our solution permits also to simplify the training process and so to reduce its required time, since the pre-training phase done on ImageNet [19] by OSVOS and RGMP is not more necessary.

Table 8.10: Results of Evaluation 4. Comparison of Siam-U-Net performance against the state-of-the-art methods, OSVOS and RGMP, in the temporal tracking setting.

Split	Siam-U-Net average DSC	OSVOS average DSC	RGMP average DSC
1	0.74 ± 0.15	0.50 ± 0.30	0.24 ± 0.29
2	0.69 ± 0.20	0.43 ± 0.27	0.53 ± 0.24
3	0.69 ± 0.16	0.45 ± 0.27	0.49 ± 0.20
4	0.69 ± 0.17	0.45 ± 0.28	0.55 ± 0.23
5	0.73 ± 0.14	0.44 ± 0.27	0.51 ± 0.28
6	0.69 ± 0.15	0.50 ± 0.26	0.49 ± 0.25
Total	0.70 ± 0.16	0.46 ± 0.28	0.47 ± 0.25

8.5.1 Limitations and Future Work

One of the main drawbacks of this study is the processing of 2D US images. An experienced clinician, when provided with volumetric data, usually exploits the information contained in neighbouring slices to interpret a 2D image. Siam-U-Net does not take advantage of this process, which has the potential to include more information and consequently allow a more accurate tracking of the cartilage. In the future, it could be interesting to adapt Siam-U-Net to work with 3D+time data, by combining a volumetric segmentation model like V-Net [288] with the siamese tracking framework.

By a qualitative evaluation of Siam-U-Net's failure cases, we discovered some situations like shown in Figure 8.15. In these cases, the upper hyperechoic line of the cartilage is not clearly defined and causes Siam-U-Net to produce segmentations where similar cartilage patterns are present (in the area identified by the mid-left green segmentation of Figure 8.15). Since this wrong output becomes the input for next step, the error could be ulteriorly propagated. To resolve these circumstances, since Siam-U-Net utilizes dropout layers, we could investigate the implementation of uncertainty estimations, in a similar fashion as done by [302]. In the best case, with an high rate of segmentation uncertainty, Siam-U-Net could integrate some mechanism to ask for reinitialisation.

An in-depth analysis of the architectural components and the tracking capabilities of our proposed solution is a valuable reference for SNN-based trackers that we are planning to work in the next future. Another interesting future direction is the adaption of Siam-U-Net for user-aided segmentation of 3D volumes (US, CT, MRI).

From a clinical point of view, the acquired US data represents several possible scenarios in robotic knee arthroscopy, but not all of them. In particular, in this proof-of-concept study the most difficult and critical situations were replicated. Future studies will include temporally longer sequences and more angles of knee flexion. Furthermore, differently from the actual surgery, the image acquisition has been performed in water. In the future, a coupling device needs to be developed to avoid the presence of air gaps at the interface between the probe and the knee surface.

8.6 Conclusions

In this chapter, we tackled a particular application: the tracking of the femoral condyle cartilage in US videos. Such an anatomical structure is one of the most at risk during MIPs, and requires segmentation masks to precisely locate its position and shape in order to avoid its damage. We demonstrated the feasibility of using a novel deep learning

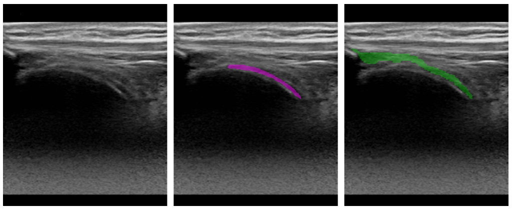


Figure 8.15: A failure example of Siam-U-Net (depicted in green in the right image). In the left US image, it can be seen that the upper hyperechoic line of the cartilage is not clearly defined.

architecture, named Siam-U-Net, to track the cartilage in real-time under simulated surgical conditions. Siam-U-Net is the combination of neural networks for medical image segmentation and the siamese framework for visual tracking. We evaluated the proposed solution using the DSC against an expert surgeon and we obtained an average performance of 0.70 ± 0.16 in the temporal tracking setting. We also present experimental results for a spatio-temporal tracking setting, showing that our solution is robust to the high variability of the cartilage appearance under the considered conditions. The high intra-operator variability (intra-operator DSC of 0.63 ± 0.30 and 0.61 ± 0.25) suggests that there are some limitations in the maximum performance that can be achieved by the network. This can be attributed to the uncertainty in the ground-truth segmentations that is dependent to the physics of the US beam. Regarding the processing speed, our network is able to run at 90 slices-per-second on a GPU-provided machine. Given its speed and accuracy, we believe that Siam-U-Net has the potential for guiding surgeons or future autonomous robotic systems during MIPs.

9 Conclusions and Future Work

This Thesis focused on visual object tracking, a fundamental problem in the field of computer vision. Particularly, the materials introduced in this manuscript tried to address particular questions that arose in the usage of deep learning methodologies to tackle visual tracking problems.

The first chapter dealt with the inefficiency of the two-stage learning procedure employed by those methodologies that used reinforcement learning to optimize deep neural networks for visual tracking. The proposed contribution, which is based on concepts of imitation learning, substitutes the learning procedure with a single end-to-end learning strategy making the learning of a tracking policy easier and more effective. Through such a learning strategy we developed two novel trackers, A3CT, which exploits demonstrations of a state-of-the-art tracker to learn an effective tracking policy, and A3CTD, that takes advantage of the same expert tracker to correct its behaviour during tracking. An extensive experimental validation on the most popular visual object tracking benchmarks showed that the proposed trackers compete with state-of-the-art solutions while running in real-time.

We then generalized the idea of learning tracking from another tracker inside a new deep learning-based framework that explicitly considers other trackers as sources of information. This framework aimed to unify application objectives such as the fast processing speed, accurate online adaptation, and fusion of trackers, that were tackled independently in the past. Our proposed solution presents a compact student model represented by a deep neural network and trained via the marriage of knowledge distillation and reinforcement learning. The first strategy allows to transfer and compress the tracking knowledge of other trackers. The second scheme enables the model to learn evaluation measures which are then exploited online. After the learning process is complete, the student can be ultimately used to build (i) a very fast single-shot tracker, (ii) a tracker with a simple and effective online adaptation mechanism, (iii) a tracker that performs fusion of other trackers. We performed an extensive validation campaign which revealed that the proposed algorithms compete with real-time state-of-the-art trackers. Despite these good results, our proposed framework was designed to work for the particular class of trackers based on deep regression networks. Future work should be devoted to the extension of the concepts of knowledge distillation and reinforcement learning to other kinds of learning architectures for tracking, such as siamese trackers, deep discriminative trackers, or transformer-based trackers.

After the discussion of the importance of building upon the knowledge of multiple and complementary trackers to achieve improved tracking performance for short-term tracking scenarios (Chapter 3), the next chapter dealt with a similar problem but in the context of long-term visual tracking. A deep learning methodology to fuse the characteristics of two complementary state-of-the-art trackers was described. Our strategy perceives whether the two trackers are following the object of interest through an online learned deep verification model. Based on this evaluation, a decision strategy that selects the best performing tracker as well as corrects the performance of the failing one is activated. The proposed solution was compared with several baselines and it was shown to beat the state-of-the-art on two popular long-term visual tracking benchmarks. Additionally, the solution was awarded as the best long-term tracker by the Visual Object Tracking Challenge VOT2021. Despite the good results achieved, further investigations should be carried out in the future. The consistency of our solution should be analyzed by considering different underlying tracker instances as well as with a number of trackers greater than two. Furthermore, to improve efficiency, new directions could explore the fusion of trackers in the image feature space rather than the state space as it was done in our study.

Chapter 5 was dedicated to mitigating the issues of domain shift and overfitting in the context of deep learning-based tracking applications. Deep regression trackers have been targeted in this study because of their fast processing speed which makes them suitable for real-time applications. Despite their efficiency, their accuracy is inadequate in many domains due to the before mentioned issues. The presented solution overcomes them by a domain adaptation strategy. This was the first methodology of such a kind developed for such a class of trackers and in the context of visual tracking. To reduce the labeling effort we proposed a weakly-supervised adaptation strategy based on the tracking-by-trackers framework (Chapter 3). In this case, reinforcement learning is used to express weak supervision as a scalar application-dependent and temporally-delayed feedback. At the same time, knowledge distillation is employed to guarantee learning stability and to transfer beneficial knowledge from more powerful but slower trackers. Extensive experiments on five different robotic vision domains demonstrated the relevance of our methodology. Real-time speed was achieved on embedded devices and on machines without GPUs, while accuracy reached significant results. We believe the problem of domain adaptation for deep learning trackers to be very relevant and that it should be taken more into consideration by the research community. We think it would be interesting to target other deep learning-based tracking methodologies for domain adaptation, as well as to develop new and better adaption strategies based on limited quantities of labeled data.

The next study focused on a domain which resulted challenging for visual trackers, deep learning-based and non: First Person Vision (FPV). In this domain, understanding human-object interactions is fundamental and tracking algorithms that follow the objects manipulated by the camera wearer can provide useful cues about what is going on in the scene. Despite a few previous attempts to exploit visual tracking in FPV tasks, a methodical analysis of the performance of state-of-the-art trackers in this domain was missing. We hence tried to fill such a gap by presenting the first systematic study of visual object tracking in FPV. Our investigation extensively analyses the performance of different visual trackers including those based on traditional methods and deep learning, and FPV-specific baseline trackers. The analysis is performed with respect to different aspects, new performance measures, and FPV-specific application requirements. This is achieved through the introduction of TREK-150, a novel benchmark dataset composed of 150 densely annotated video sequences. Our results showed that object tracking in FPV is challenging, and suggested that more research efforts are needed for this problem so that tracking could benefit FPV tasks. Particularly, we think our study could open up many new research directions at the intersection of FPV and visual tracking. Indeed, our outcomes revealed that there is a need of developing more accurate trackers for this domain, which we think will require new principles. Moreover, considering the importance of object motion information in many different FPV application tasks, we think it would be interesting to understand which is the contribution brought to such particular applications by the many different tracking approaches available today. Such an impact could be evaluated by measuring the correlation between the results obtained with standard metrics used to evaluate the performance of trackers and the results achieved by the metrics used to evaluate downstream algorithms that employ trackers.

All the studies presented in the previous chapters assumed that bounding-boxes were employed to represent the targets' states. In the next chapter, we tried to move away from such a representation. We presented an extensive exploration of deep learning-based segmentation methods available in the computer vision community that can be conditioned on the target to be tracked. Such segmentation methods were used to provide target segmentation after the output of bounding-box trackers. By this strategy, any bounding-box tracker can be transformed into a segmentation tracker. Our analysis demonstrated that the proposed combination allows bounding-box trackers to compete with recently proposed segmentation trackers while performing quasi real-time.

Finally, the last chapter studied a particular tracking problem in which precise information regarding the position and shape of the target in the form of a segmentation mask is required: the tracking of the knee cartilage during ultrasound-guided minimally invasive procedures. A new deep learning method has been presented to track, accurately and efficiently, the femoral condyle cartilage in ultrasound sequences acquired under several clinical conditions, mimicking realistic surgical setups. The proposed solution combines a deep learning segmentation architecture with the siamese framework to track the cartilage in temporal and spatio-temporal sequences of 2D ultrasound images. Through extensive performance validation, we demonstrated that our algorithm is able to track the femoral condyle cartilage with an accuracy comparable to experienced surgeons. It was additionally shown that the proposed method outperforms state-of-the-art segmentation models and trackers in the localization of such anatomical structure. Given these outcomes, we claim that the proposed solution has the potential for ultrasound guidance in minimally invasive knee procedures. And we expect further efficacy and efficiency of the solution to be achieved by tracking methodologies exploiting the 4D information contained in 3D ultrasound scans.

Publications

Part of the material presented in this Thesis appeared (or will appear) in the following publications.

1. **Matteo Dunnhofer**, Niki Martinel, Gian Luca Foresti, Christian Micheloni, "Visual Tracking by Means of Deep Reinforcement Learning and an Expert Demonstrator", *2019 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019, pp. 2290-2299, doi: 10.1109/ICCVW.2019.00282.

2. Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Cehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, Abdelrahman Eldesokey, Jani Kapyla, Gustavo Fernandez, ..., **Matteo Dunnhofer**, ..., "The Seventh Visual Object Tracking VOT2019 Challenge Results", *2019 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019, pp. 2206-2241, doi: 10.1109/ICCVW.2019.00276.

3. **Matteo Dunnhofer**, Maria Antico, Fumio Sasazawa, Yu Takeda, Saskia Camps, Niki Martinel, Christian Micheloni, Gustavo Carneiro, Davide Fontanarosa, "Siam-U-Net: encoder-decoder siamese network for knee cartilage tracking in ultrasound images", *Medical Image Analysis*, Volume 60, 101631, 2020, doi: 10.1016/j.media.2019.101631.

4. **Matteo Dunnhofer**, Niki Martinel, Christian Micheloni, "Tracking-by-Trackers with a Distilled and Reinforced Model", *Computer Vision – ACCV 2020*, 2021, Springer International Publishing

5. **Matteo Dunnhofer**, Niki Martinel, Christian Micheloni, "An Exploration of Target-Conditioned Segmentation Methods for Visual Object Trackers", *Computer Vision – ECCV 2020 Workshops*, 2020, Springer International Publishing

6. Matej Kristan, Ales Leonardis, Jirí Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Cehovin Zajc, Alan Lukezic, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernández, ..., **Matteo Dunnhofer**, ..., "The Eight Visual Object Tracking VOT2020 Challenge Results", *Computer Vision – ECCV 2020 Workshops*, 2020, Springer International Publishing

7. **Matteo Dunnhofer**, Niki Martinel, Christian Micheloni, "Weakly-Supervised Domain Adaptation of Deep Regression Trackers via Reinforced Knowledge Distillation", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5016-5023, July 2021, doi: 10.1109/LRA.2021.3070816.

8. **Matteo Dunnhofer**, Antonino Furnari, Giovanni Maria Farinella, Christian Micheloni, “Is First Person Vision Challenging for Object Tracking?”, *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*

9. Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Cehovin, Alan Lukežič, Ondrej Drbohlav, Jani Käpylä, Gustav Häger, Song Yan, Jinyu Yang, Zhongqun Zhang, Gustavo Fernández, ..., **Matteo Dunnhofer**, ..., “The Ninth Visual Object Tracking VOT2021 Challenge Results”, *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*

10. **Matteo Dunnhofer**, Antonino Furnari, Giovanni Maria Farinella, Christian Micheloni, “Visual Object Tracking in First Person Vision”, *submitted to a journal*, 2022

11. **Matteo Dunnhofer**, Christian Micheloni, “CoCoLoT: Combining Complementary Trackers for in Long-Term Visual Tracking”, *submitted to a conference*, 2022

12. **Matteo Dunnhofer**, Kristian Simonato, Christian Micheloni, “Combining Complementary Trackers for Enhanced Long-Term Visual Object Tracking”, *submitted to a journal*, 2022

I would like to acknowledge the EU-funded H2020 project “ACHIEVE-ITN” for having supported most of the studies presented in this Thesis.

Bibliography

[1] Jamie Enoch, Leanne McDonald, Lee Jones, Pete R Jones, and David P Crabb. Evaluating whether sight is the most valued sense. *JAMA ophthalmology*, 137(11):1317–1320, 2019.

[2] Michael K. Tanenhaus and Michael J. Spivey-Knowlton. Eye-tracking. *Language and Cognitive Processes*, 11(6):583–588, 1996.

[3] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13–es, December 2006.

[4] Arnold W. M. Smeulders, Dung M. Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[5] Dr Emilio Maggio and Dr Andrea Cavallaro. *Video Tracking: Theory and Practice.* Wiley Publishing, 1st edition, 2011.

[6] Alan Lukežič, Tomáš Vojíř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *International Journal of Computer Vision*, 126(7):671–688, 2018.

[7] Jan Salomon Cramer. The origins of logistic regression. 2002.

[8] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[9] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[10] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[11] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry.* MIT press, 2017.

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[14] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[15] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[16] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354—-, 2017.

[17] James B Heaton, Nick G Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, jun 2009.

[20] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[21] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

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

[23] A. Goshtasby, S. H. Gage, and J. F. Bartholic. A two-stage cross correlation approach to template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(3):374–378, 1984.

[24] J.P. Lewis. Fast template matching. *Vis. Interface*, 95, 11 1994.

[25] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:142–149, 2000.

[26] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[27] M. Swain and D. Ballard. Indexing via color histograms. In *Proceedings Third International Conference on Computer Vision*, pages 390,391,392,393, Los Alamitos, CA, USA, dec 1990. IEEE Computer Society.

[28] Mario Edoardo Maresca and Alfredo Petrosino. MATRIOSKA: A multi-level approach to fast tracking by learning. In *International Conference on Image Analysis and Processing*, volume 8157 LNCS, pages 419–428, 2013.

[29] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):941–953, 2013.

[30] Hyeonseob Nam, Seunghoon Hong, and Bohyung Han. Online graph-based tracking. In *European Conference on Computer Vision*, volume 8693 LNCS, pages 112–126. Springer Verlag, 2014.

[31] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010.

[32] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[33] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative Scale Space Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, 2017.

[34] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H.S. Torr. Staple: Complementary learners for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 1401–1409, 2016.

[35] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference*, pages 6.1–6.10. BMVA Press, 2006. doi:10.5244/C.20.6.

[36] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.

[37] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. In *European Conference on Computer Vision*, volume 11205 LNCS, pages 310–327. Springer Verlag, mar 2018.

[38] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A High-quality Benchmark for Large-scale Single Object Tracking. In *International Conference on Computer Vision and Pattern Recognition*, sep 2019.

[39] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. oct 2018.

[40] Hyeonseob Nam and Bohyung Han. Learning Multi-domain Convolutional Neural Networks for Visual Tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016-Decem:4293–4302, 2016.

[41] David Held, Sebastian Thrun, and Silvio Savarese. Learning to Track at 100 FPS with Deep Regression Networks. In *European Conference on Computer Vision*, volume abs/1604.0, 2016.

[42] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re 3 : Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018.

[43] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H.S. Torr. Fully-convolutional siamese networks for object tracking. *European Conference on Computer Vision*, 9914 LNCS:850–865, 2016.

[44] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High Performance Visual Tracking with Siamese Region Proposal Network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980. IEEE, jun 2018.

[45] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SIAMRPN++: Evolution of siamese visual tracking with very deep networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019-June:4277–4286, 2019.

[46] Zhipeng Zhang and Houwen Peng. Deeper and Wider Siamese Networks for Real-Time Visual Tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, jan 2019.

[47] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H S Torr. Fast Online Object Tracking and Segmentation: A Unifying Approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[48] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[49] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A Riedmiller. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5, 2013.

[50] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1349–1358. IEEE, jul 2017.

[51] James Steven Supancic III and Deva Ramanan. Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning. *CoRR*, abs/1707.0, 2017.

[52] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Visual Tracking by Reinforced Decision Making. *CoRR*, abs/1702.0, 2017.

[53] Liangliang Ren, Xin Yuan, Jiwen Lu, Ming Yang, and Jie Zhou. Deep Reinforcement Learning with Iterative Shift for Visual Tracking. In *The European Conference on Computer Vision (ECCV)*, 2018.

[54] Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu. Real-time 'Actor-Critic' Tracking. In *The European Conference on Computer Vision (ECCV)*, 2018.

[55] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, USA, 2nd edition, 2018.

[56] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[57] Ronald J Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8(3-4):229–256, may 1992.

[58] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

[59] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In *European Conference on Computer Vision*, aug 2016.

[60] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient Convolution Operators for Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, nov 2017.

[61] Martin Danelljan, Goutam Bhat, Fahad Khan, and Michael Felsberg. ATOM: Accurate Tracking by Overlap Maximization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[62] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning Discriminative Model Prediction for Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[63] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *CoRR*, abs/2004.05439, 2020.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[65] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1571–1580, June 2021.

[66] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8126–8135, June 2021.

[67] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV*, 2021.

[68] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson Lau, and Ming-Hsuan Yang. CREST: Convolutional Residual Learning for Visual Tracking. In *Internationaò Conference on Computer Vision*, aug 2017.

[69] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *CoRR*, abs/1602.0, 2016.

[70] Matthias Mueller, Neil Smith, and Bernard Ghanem. A Benchmark and Simulator for UAV Tracking. In *European Conference on Computer Vision*, pages 445–461. Springer, Cham, 2016.

[71] Yi Wu, Jongwoo Lim, and Ming Hsuan Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418. IEEE Computer Society, 2013.

[72] Matej Kristan, Aleš Leonardis, Jiři Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin, Tomáš Vojír, Gustav Häger, Alan Lukežič, Gustavo Fernández, et al. The Visual Object Tracking VOT2016 Challenge Results. In *European Conference on Computer Vision*, pages 777–823. Springer, Cham, 2016.

[73] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin Zajc, Tomáš Vojír, Goutam Bhat, Alan Lukežič, Abdelrahman Eldesokey, Gustavo Fernández, et al. The Sixth Visual Object Tracking VOT2018 Challenge Results. In *European Conference on Computer Vision*, pages 3–53. Springer, Cham, sep 2019.

[74] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of {Go} with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.

[75] Christopher J C H Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[76] Vijay R Konda and John N Tsitsiklis. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, 2000.

[77] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. jul 2017.

[78] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6292–6299. Institute of Electrical and Electronics Engineers Inc., sep 2018.

[79] Tim Salimans and Richard Chen. Learning Montezuma's Revenge from a Single Demonstration. In *Proceedings of NeurIPS 2018 Workshop on Deep RL*, dec 2018.

[80] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy Optimization with Demonstrations. In *ICML 2018*, volume 80, pages 2474–2483, 2018.

[81] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from Demonstrations. In *AAAI*, apr 2018.

[82] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomáš Vojíř, Gustavo Fernández, Alan Lukežič, et al. The Visual Object Tracking VOT2014 Challenge Results. In *European Conference on Computer Vision*, pages 191–217. Springer, Cham, 2015.

[83] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, Roman Pflugfelder, et al. The Visual Object Tracking VOT2015 Challenge Results. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 564–586. IEEE, dec 2015.

[84] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, et al. The Visual Object Tracking VOT2017 Challenge Results. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1949–1972. IEEE, oct 2017.

[85] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning Dynamic Siamese Network for Visual Object Tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1781–1789. IEEE, oct 2017.

[86] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision*, volume 11213 LNCS, pages 103–119, aug 2018.

[87] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[88] Chen Huang, Simon Lucey, and Deva Ramanan. Learning Policies for Adaptive Tracking with Deep Feature Cascades. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 105–114. IEEE, oct 2017.

[89] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, New York, New York, USA, 2009. ACM Press.

[90] Diederik P Kingma and Jimmy Ba. Adam: {A} Method for Stochastic Optimization. *CoRR*, abs/1412.6, 2014.

[91] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[92] Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, and et al. The Seventh Visual Object Tracking VOT2019 Challenge Results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[93] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-End Representation Learning for Correlation Filter Based Tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5000–5008. IEEE, jul 2017.

[94] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernández, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016.

[95] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Learning Adaptive Discriminative Correlation Filters via Temporal Consistency Preserving Spatial Feature Selection for Robust Visual Tracking. *IEEE Transactions on Image Processing*, jul 2019.

[96] J. Shen, X. Tang, X. Dong, and L. Shao. Visual object tracking by hierarchical attention siamese network. *IEEE Transactions on Cybernetics*, 50(7):3068–3080, 2020.

[97] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[98] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware Anchor-free Tracking. In *European Conference on Computer Vision*, jun 2020.

[99] Jianming Zhang, Shugao Ma, and Stan Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *European Conference on Computer Vision*, volume 8694 LNCS, pages 188–203. Springer Verlag, 2014.

[100] Christian Bailer, Alain Pagani, and Didier Stricker. A superior tracking approach: Building a strong tracker through fusion. In *European Conference on Computer Vision*, volume 8695 LNCS, pages 170–185. Springer Verlag, 2014.

[101] Tomas Vojir, Jiri Matas, and Jana Noskova. Online adaptive hidden Markov model for multi-tracker fusion. *Computer Vision and Image Understanding*, 153:109–119, dec 2016.

[102] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *Deep Learning Workshop NIPS 2014*, mar 2014.

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778, 2016.

[104] Zhiyuan Tang, Dong Wang, and Zhiyong Zhang. Recurrent neural network training with dark knowledge transfer. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2016-May, pages 5900–5904, may 2016.

[105] Krzysztof J. Geras, Abdel-rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton. Blending LSTMs into CNNs. nov 2015.

[106] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, feb 2018.

[107] W. Wang, K. Zhang, M. Lv, and J. Wang. Hierarchical spatiotemporal context-aware correlation filters for visual tracking. *IEEE Transactions on Cybernetics*, pages 1–14, 2020.

[108] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming Hsuan Yang. Hedged Deep Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 4303–4311, 2016.

[109] Zhetao Li, Wei Wei, Tianzhu Zhang, Meng Wang, Sujuan Hou, and Xin Peng. Online Multi-Expert Learning for Visual Tracking. *IEEE Transactions on Image Processing*, 29:934–946, 2019.

[110] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic deep multitask and transfer reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016*, nov 2016.

[111] H. Zhao, X. Sun, J. Dong, C. Chen, and Z. Dong. Highlight every step: Knowledge distillation via collaborative teaching. *IEEE Transactions on Cybernetics*, pages 1–12, 2020.

[112] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 743–752, 2017.

[113] Tong He, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan. Knowledge Adaptation for Efficient Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 578–587, 2019.

[114] Y. Zhang, X. Li, and Z. Zhang. Efficient person search via expert-guided knowledge distillation. *IEEE Transactions on Cybernetics*, pages 1–12, 2019.

[115] Ning Wang, Wengang Zhou, Yibing Song, Chao Ma, and Houqiang Li. Real-Time Correlation Tracking Via Joint Model Compression and Transfer. *IEEE Transactions on Image Processing*, 29:6123–6135, jul 2020.

[116] Yuanpei Liu, Xingping Dong, Xiankai Lu, Fahad Shahbaz Khan, Jianbing Shen, and Steven Hoi. Teacher-Students Knowledge Distillation for Siamese Trackers. jul 2019.

[117] Qing Guo, Ruize Han, Wei Feng, Zhihao Chen, and Liang Wan. Selective Spatial Regularization by Reinforcement Learned Decision Making for Object Tracking. *IEEE Transactions on Image Processing*, 29:2999–3013, 2020.

[118] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Boron Yotam, Firoiu Vlad, Harley Tim, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *35th International Conference on Machine Learning, ICML 2018*, volume 4, pages 2263–2284, feb 2018.

[119] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, et al. The eighth visual object tracking vot2020 challenge results. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 547–601, Cham, 2020. Springer International Publishing.

[120] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the Variance of the Adaptive Learning Rate and Beyond. aug 2019.

[121] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 4793–4801. Institute of Electrical and Electronics Engineers Inc., oct 2019.

[122] Tianyu Yang, Pengfei Xu, Runbo Hu, Hua Chai, and Antoni B. Chan. ROAM: Recurrently Optimizing Tracking Model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, jul 2020.

[123] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, nov 2020.

[124] Alan Lukezic, Jiri Matas, and Matej Kristan. D3s - a discriminative single shot segmentation tracker. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[125] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. Graph Convolutional Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, number 1, pages 4649–4659, 2019.

[126] Paul Voigtlaender, Jonathon Luiten, Philip H.S. Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[127] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[128] Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. GradNet: Gradient-Guided Network for Visual Object Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[129] Bingyan Liao, Chenye Wang, Yayun Wang, Yaonong Wang, and Jun Yin. Pg-net: Pixel to global matching network for visual tracking. In *16th European Conference on Computer Vision (ECCV) 2020*, volume 12367 of *Lecture Notes in Computer Science*, pages 429–444. Springer, 2020.

[130] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know Your Surroundings: Exploiting Scene Information for Object Tracking. In *European Conference on Computer Vision*, mar 2020.

[131] Matteo Dunnhofer, Niki Martinel, and Christian Micheloni. An exploration of target-conditioned segmentation methods for visual object trackers. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 618–636, Cham, 2020. Springer International Publishing.

[132] Eunbyung Park and Alexander C. Berg. Meta-tracker: Fast and Robust Online Adaptation for Visual Object Trackers. In *European Conference on Computer Vision*, volume 11207 LNCS, pages 587–604. Springer Verlag, jan 2018.

[133] Alan Lukežič, Luka Čehovin Zajc, Tomáš Vojíř, Jiří Matas, and Matej Kristan. Performance evaluation methodology for long-term single-object tracking. *IEEE Transactions on Cybernetics*, pages 1–14, 2020.

[134] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding Color Information for Visual Tracking: Algorithms and Benchmark. *IEEE Transactions on Image Processing*, 24(12):5630–5644, dec 2015.

[135] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for Speed: A Benchmark for Higher Frame Rate Object Tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 1134–1143. Institute of Electrical and Electronics Engineers Inc., mar 2017.

[136] Dongyan Guo, Yanyan Shao, Ying Cui, Zhenhua Wang, Liyan Zhang, and Chunhua Shen. Graph attention tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9543–9552, June 2021.

[137] Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-Performance Long-Term Tracking With Meta-Updater. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6297–6306. Institute of Electrical and Electronics Engineers (IEEE), aug 2020.

[138] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[139] Heng Fan and Haibin Ling. Parallel tracking and verifying. *IEEE Transactions on Image Processing*, 28(8):4130–4144, 2019.

[140] Bin Yan, Haojie Zhao, Dong Wang, Huchuan Lu, and Xiaoyun Yang. 'Skimming-perusal' tracking: A framework for real-time and robust long-term tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 2385–2393, 2019.

[141] Ju Hong Yoon, Du Yong Kim, and Kuk Jin Yoon. Visual tracking via adaptive tracker selection with multiple features. In *European Conference on Computer Vision*, volume 7575 LNCS, pages 28–41, 2012.

[142] Matteo Dunnhofer, Niki Martinel, and Christian Micheloni. Tracking-by-Trackers with a Distilled and Reinforced Model. In *Asian Conference on Computer Vision*, 2020.

[143] Abhinav Moudgil and Vineet Gandhi. Long-term visual object tracking benchmark. In *Asian Conference on Computer Vision*, pages 629–645. Springer, 2018.

[144] Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Cehovin, Alan Lukežič, Ondrej Drbohlav, Jani Käpylä, Gustav Häger, Song Yan, Jinyu Yang, Zhongqun Zhang, and Gustavo Fernández. The ninth visual object tracking vot2021 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2711–2738, October 2021.

[145] Alan Lukežič, Luka Čehovin Zajc, Tomáš Vojíř, Jiří Matas, and Matej Kristan. Fucolot – a fully-correlational long-term tracker. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 595–611, Cham, 2019. Springer International Publishing.

[146] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GlobalTrack: A Simple and Strong Baseline for Long-term Tracking. In *AAAI Conference on Artificial Intelligence*, dec 2020.

[147] Naiyan Wang and Dit Yan Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *31st International Conference on Machine Learning, ICML 2014*, volume 4, pages 2807–2817, 2014.

[148] ke Song, Wei Zhang, Ran Song, and Yibin Li. Online decision based visual tracking via reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11778–11788. Curran Associates, Inc., 2020.

[149] Seokeon Choi, Junhyun Lee, Yunsung Lee, and Alexander G. Hauptmann. Robust long-term object tracking via improved discriminative model prediction. volume abs/2008.04722, 2020.

[150] Zikai Zhang, Bineng Zhong, Shengping Zhang, Zhenjun Tang, Xin Liu, and Zhaoxiang Zhang. Distractor-aware fast tracking via dynamic convolutions and mot philosophy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1024–1033, 2021.

[151] Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

[152] Zitong Yi, Zhihang Tong, Yanyun Zhao, Zhicheng Zhao, and Fei Su. A method of stable long-term single object tracking. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021.

[153] Lang Yu, Huanlong Zhang, Junyang Yu, and Baojun Qiao. Online-adaptive classification and regression network with sample-efficient meta learning for long-term tracking. *Image and Vision Computing*, 112:104181, 2021.

[154] N. Papanikolopoulos, P. K. Khosla, and T. Kanada. Vision and control techniques for robotic visual tracking. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 857–864 vol.1, 1991.

[155] Jan Portmann, Simon Lynen, Margarita Chli, and Roland Siegwart. People detection and tracking from aerial thermal views. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1794–1800. Institute of Electrical and Electronics Engineers Inc., sep 2014.

[156] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, sep 2013.

[157] Florian Shkurti, Wei Di Chang, Peter Henderson, Md Jahidul Islam, Juan Camilo Gamboa Higuera, Jimmy Li, Travis Manderson, Anqi Xu, Gregory Dudek, and Junaed Sattar. Underwater multi-robot convoying using visual tracking by detection. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-Septe, pages 4189–4196. Institute of Electrical and Electronics Engineers Inc., dec 2017.

[158] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 5(2):1803–1810, apr 2020.

[159] Matteo Dunnhofer, Maria Antico, Fumio Sasazawa, Yu Takeda, Saskia Camps, Niki Martinel, Christian Micheloni, Gustavo Carneiro, and Davide Fontanarosa. Siam-U-Net: encoder-decoder siamese network for knee cartilage tracking in ultrasound images. *Medical Image Analysis*, 60:101631, feb 2020.

[160] Krishneel Chaudhary, Moju Zhao, Fan Shi, Xiangyu Chen, Kei Okada, and Masayuki Inaba. Robust real-time visual tracking using dual-frame deep comparison network integrated with correlation filters. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-Septe, pages 6837–6842. Institute of Electrical and Electronics Engineers Inc., dec 2017.

[161] Sangeeth Reddy, Minesh Mathew, Lluis Gomez, Marcal Rusinol, Dimosthenis Karatzas, and C. V. Jawahar. RoadText-1K: Text Detection Recognition Dataset for Driving Videos. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 11074–11080. Institute of Electrical and Electronics Engineers Inc., may 2020.

[162] Karin De Langis and Junaed Sattar. Realtime Multi-Diver Tracking and Reidentification for Underwater Human-Robot Collaboration. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 11140–11146. Institute of Electrical and Electronics Engineers Inc., may 2020.

[163] Ankush Roy, Xi Zhang, Nina Wolleb, Camilo Perez Quintero, and Martin Jagersand. Tracking benchmark and evaluation for manipulation tasks. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2015-June, pages 2448–2453. Institute of Electrical and Electronics Engineers Inc., jun 2015.

[164] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning, 2010.

[165] Gabriela Csurka. A comprehensive survey on domain adaptation for visual applications. In *Advances in Computer Vision and Pattern Recognition*, number 9783319583464, pages 1–35. Springer London, 2017.

[166] Gabriele Angeletti, Barbara Caputo, and Tatiana Tommasi. Adaptive Deep Learning Through Visual Domain Localization. In *Proceedings - IEEE International*

Conference on Robotics and Automation, pages 7135–7142. Institute of Electrical and Electronics Engineers Inc., sep 2018.

[167] Jingwei Zhang, Lei Tai, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. VR-Goggles for Robots: Real-to-Sim Domain Adaptation for Visual Control. *IEEE Robotics and Automation Letters*, 4(2):1148–1155, apr 2019.

[168] Alexandra Carlson, Katherine A. Skinner, Ram Vasudevan, and Matthew Johnson-Roberson. Sensor Transfer: Learning Optimal Sensor Effect Image Augmentation for Sim-to-Real Domain Adaptation. *IEEE Robotics and Automation Letters*, 4(3):2431–2438, jul 2019.

[169] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-Septe, pages 1551–1558. Institute of Electrical and Electronics Engineers Inc., dec 2017.

[170] Massimiliano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, and Barbara Caputo. Kitting in the Wild through Online Domain Adaptation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1103–1109. Institute of Electrical and Electronics Engineers Inc., dec 2018.

[171] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3516–3523. Institute of Electrical and Electronics Engineers Inc., sep 2018.

[172] Mohammad Reza Loghmani, Luca Robbiano, Mirco Planamente, Kiru Park, Barbara Caputo, and Markus Vincze. Unsupervised Domain Adaptation through Inter-Modal Rotation for RGB-D Object Recognition. *IEEE Robotics and Automation Letters*, 5(4):6631–6638, oct 2020.

[173] Enrico Bellocchio, Gabriele Costante, Silvia Cascianelli, Mario Luca Fravolini, and Paolo Valigi. Combining Domain Adaptation and Spatial Consistency for Unseen Fruits Counting: A Quasi-Unsupervised Approach. *IEEE Robotics and Automation Letters*, 5(2):1079–1086, apr 2020.

[174] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively Parallel Methods for Deep Reinforcement Learning. jul 2015.

[175] Matteo Dunnhofer, Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Visual Tracking by means of Deep Reinforcement Learning and an Expert Demonstrator. In *Proceedings of The IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[176] Dawei Du et al. VisDrone-SOT2019: The vision meets drone single object tracking challenge results. In *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 199–212, 2019.

[177] Qiao Liu, Zhenyu He, Xin Li, and Yuan Zheng. PTB-TIR: A Thermal Infrared Pedestrian Tracking Benchmark. *IEEE Transactions on Multimedia*, 22(3):666–675, mar 2020.

[178] Takeo Kanade and Martial Hebert. First-person vision. *Proceedings of the IEEE*, 100(8):2442–2453, 2012.

[179] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[180] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Symbiotic attention with privileged information for egocentric action recognition. In *AAAI Conference on Artificial Intelligence*, 2020.

[181] Miao Liu, Siyu Tang, Yin Li, and James Rehg. Forecasting human object interaction: Joint prediction of motor attention and actions in first person video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2, 2020.

[182] Antonino Furnari and Giovanni Farinella. Rolling-Unrolling LSTMs for Action Anticipation from First-Person Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, may 2020.

[183] Dima Damen, Teesid Leelasawassuk, and Walterio Mayol-Cuevas. You-do, i-learn: Egocentric unsupervised discovery of objects and their modes of interaction towards video-based guidance. *Computer Vision and Image Understanding*, 149:98–112, 2016.

[184] Francesco Ragusa, Antonino Furnari, Salvatore Livatino, and Giovanni Maria Farinella. The meccano dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain. In *IEEE Winter Conference on Application of Computer Vision (WACV)*, 2020.

[185] Minjie Cai, Kris M Kitani, and Yoichi Sato. Understanding hand-object manipulation with grasp types and object attributes. In *Robotics: Science and Systems*, volume 3. Ann Arbor, Michigan;, 2016.

[186] Gedas Bertasius, Hyun Soo Park, Stella X. Yu, and Jianbo Shi. First-person action-object detection with egonet. In *Proceedings of Robotics: Science and Systems*, July 2017.

[187] Gedas Bertasius, Hyun Soo Park, Stella X Yu, and Jianbo Shi. Unsupervised learning of important objects from first-person videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1956–1964, 2017.

[188] Zhe Cao, Ilija Radosavovic, Angjoo Kanazawa, and Jitendra Malik. Reconstructing hand-object interactions in the wild. *arXiv preprint arXiv:2012.09856*, 2020.

[189] Maedeh Aghaei, Mariella Dimiccoli, and Petia Radeva. Multi-face tracking by extended bag-of-tracklets in egocentric photo-streams. *Computer Vision and Image Understanding*, 149:146–156, aug 2016.

[190] Maedeh Aghaei, Mariella Dimiccoli, and Petia Radeva. With whom do i interact? Detecting social interactions in egocentric photo-streams. In *Proceedings - International Conference on Pattern Recognition*, volume 0, pages 2959–2964. Institute of Electrical and Electronics Engineers Inc., jan 2016.

[191] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, et al. Ego4d: Around the World in 3,000 Hours of Egocentric Video. *CoRR*, abs/2110.07058, 2021.

[192] Ryan J. Visee, Jirapat Likitlersuang, and Jose Zariffa. An Effective and Efficient Method for Detecting Hands in Egocentric Videos for Rehabilitation Applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(3):748–755, mar 2020.

[193] Georgios Kapidis, Ronald Poppe, Elsbeth Van Dam, Lucas Noldus, and Remco Veltkamp. Egocentric hand track and object-based human action recognition. In *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 922–929. IEEE, 2019.

[194] Stefano Alletto, Giuseppe Serra, and Rita Cucchiara. Egocentric object tracking: an odometry-based solution. In *International Conference on Image Analysis and Processing*, pages 687–696. Springer, 2015.

[195] Jyoti Nigam and Renu M Rameshan. EgoTracker: Pedestrian Tracking with Reidentification in Egocentric Videos. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2017-July, pages 980–987, 2017.

[196] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-Time Hand Tracking Under Occlusion from an Egocentric RGB-D Sensor. In *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, volume 2018-Janua, pages 1284–1293, 2017.

[197] Xianjing Han, Xuemeng Song, Yiyang Yao, Xin Shun Xu, and Liqiang Nie. Neural Compatibility Modeling with Probabilistic Knowledge Distillation. *IEEE Transactions on Image Processing*, 29:871–882, 2020.

[198] Li Sun, Ulrich Klank, and Michael Beetz. EYEWATCHME-3D Hand and object tracking for inside out activity analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16. Institute of Electrical and Electronics Engineers (IEEE), mar 2010.

[199] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 284–293, 2019.

[200] Minghuang Ma, Haoqi Fan, and Kris M Kitani. Going deeper into first-person activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1894–1903, 2016.

[201] Ivan Rodin, Antonino Furnari, Dimitrios Mavroedis, and Giovanni Maria Farinella. Predicting the future from first person (egocentric) vision: A survey. *Computer Vision and Image Understanding*, 2021.

[202] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long-range video understanding. In *European Conference on Computer Vision*, pages 154–171. Springer, 2020.

[203] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, pages 1–23, 2021.

[204] Annan Li, Min Lin, Yi Wu, Ming Hsuan Yang, and Shuicheng Yan. NUS-PRO: A New Visual Tracking Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):335–349, feb 2016.

[205] Abhinav Rai, Fadime Sener, and Angela Yao. Transformed rois for capturing visual transformations in videos. *CoRR*, abs/2106.03162, 2021.

[206] Alan Lukezic, Ugur Kart, Jani Kapyla, Ahmed Durmush, Joni Kristian Kamarainen, Jiri Matas, and Matej Kristan. CDTB: A color and depth visual object tracking dataset and benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 10012–10021. Microsoft Research Asia, jul 2019.

[207] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson W.H. Lau, and Ming Hsuan Yang. VITAL: VIsual Tracking via Adversarial Learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8990–8999. IEEE Computer Society, apr 2018.

[208] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming Ming Cheng, Stephen L Hicks, and Philip H.S. Torr. Struck: Structured Output Tracking with Kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016.

[209] Joseph Redmon, Santosh Kumar Divvala, Ross B Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[210] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *Proceedings - International Conference on Image Processing, ICIP*, volume 2017-Septe, pages 3645–3649. IEEE Computer Society, mar 2018.

[211] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning Background-Aware Correlation Filters for Visual Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[212] Mohamed H. Abdelpakey and Mohamed S. Shehata. DP-Siam: Dynamic Policy Siamese Network for Robust Object Tracking. *IEEE Transactions on Image Processing*, 29:1479–1492, 2020.

[213] Zhiyuan Liang and Jianbing Shen. Local Semantic Siamese Networks for Fast Tracking. *IEEE Transactions on Image Processing*, 29:3351–3364, 2020.

[214] Wenzhang Zhou, Longyin Wen, Libo Zhang, Dawei Du, Tiejian Luo, and Yanjun Wu. Siamcan: Real-time visual tracking based on siamese center-aware network. *IEEE Transactions on Image Processing*, 30:3597–3609, 2021.

[215] Yi Wu, Jongwoo Lim, and Ming Hsuan Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418. IEEE Computer Society, 2013.

[216] Ricardo Sanchez-Matilla and Andrea Cavallaro. A predictor of moving objects for first-person vision. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2189–2193. IEEE, 2019.

[217] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2847–2854. IEEE, 2012.

[218] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge: A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision*, 129(4):845–881, 2021.

[219] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129(2):548–578, 2021.

[220] Jack Valmadre, Luca Bertinetto, João F. Henriques, Ran Tao, Andrea Vedaldi, Arnold W.M. Smeulders, Philip H.S. Torr, and Efstratios Gavves. Long-Term Tracking in the Wild: A Benchmark. In *European Conference on Computer Vision*, volume 11207 LNCS, pages 692–707. Springer Verlag, mar 2018.

[221] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Maria Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 49:401–411, nov 2017.

[222] Antonino Furnari and Giovanni Maria Farinella. What Would You Expect? Anticipating Egocentric Actions With Rolling-Unrolling LSTMs and Modality Attention. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[223] Georgios Kapidis, Ronald Poppe, Elsbeth Van Dam, Lucas Noldus, and Remco Veltkamp. Egocentric hand track and object-based human action recognition. In *Proceedings - 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, pages 922–929. Institute of Electrical and Electronics Engineers Inc., may 2019.

[224] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F. Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[225] Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming Hsuan Yang. Learning Spatial-Temporal Regularized Correlation Filters for Visual Tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4904–4913. IEEE Computer Society, mar 2018.

[226] Ning Wang, Wengang Zhou, Qi Tian, Richang Hong, Meng Wang, and Houqiang Li. Multi-cue Correlation Filters for Robust Visual Tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4844–4853, 2018.

[227] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines. In *AAAI Conference on Artificial Intelligence*, nov 2020.

[228] Zhihong Fu, Qingjie Liu, Zehua Fu, and Yunhong Wang. Stmtrack: Template-free visual tracking with space-time memory networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13774–13783, 2021.

[229] Alan Lukežič, Luka Čehovin Zajc, Tomáš Vojíř, Jiří Matas, and Matej Kristan. Now you see me: evaluating performance in long-term visual tracking, 2018.

[230] Qiang Wang, Jin Gao, Junliang Xing, Mengdan Zhang, and Weiming Hu. DCFNet: Discriminant Correlation Filters Network for Visual Tracking. apr 2017.

[231] Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming Hsuan Yang. Deep regression tracking with shrinkage loss. In *European Conference on Computer Vision*, volume 11218 LNCS, pages 369–386, 2018.

[232] Lichao Zhang, Abel Gonzalez-Garcia, Joost Van De Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 4009–4018. Institute of Electrical and Electronics Engineers Inc., oct 2019.

[233] Bin Yan, Houwen Peng, Kan Wu, Dong Wang, Jianlong Fu, and Huchuan Lu. Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15180–15189, June 2021.

[234] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 473–488, Cham, 2020. Springer International Publishing.

[235] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[236] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-Time MDNet. In *European Conference on Computer Vision*, 2018.

[237] Olga Russakovsky, Li Jia Li, and Li Fei-Fei. Best of both worlds: Human-machine collaboration for object annotation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[238] Fuxing Yang, Michael A. Mackey, Fiorenza Ianzini, Greg Gallardo, and Milan Sonka. Cell segmentation, tracking, and mitosis detection using temporal context. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, oct 2005.

[239] Min Li, Chandra Kambhamettu, and Maureen Stone. Automatic contour tracking in ultrasound images. *Clinical Linguistics and Phonetics*, sep 2005.

[240] N. J.B. McFarlane and C. P. Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, may 1995.

[241] Changick Kim and Jenq Neng Hwang. Fast and automatic video object segmentation and tracking for content-based applications. *IEEE Transactions on Circuits and Systems for Video Technology*, feb 2002.

[242] S. Caelles, K. K. Maninis, J. Pont-Tuset, L. Leal-Taixe, D. Cremers, and L. Van Gool. One-Shot Video Object Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5320–5329. IEEE, jul 2017.

[243] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast Video Object Segmentation by Reference-Guided Mask Propagation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[244] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *British Machine Vision Conference 2017*. BMVA Press, jun 2017.

[245] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. PReMVOS: Proposal-Generation, Refinement and Merging for Video Object Segmentation. In *Asian Conference on Computer Vision*. Springer Verlag, jul 2018.

[246] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K. Katsaggelos. Efficient Video Object Segmentation via Network Modulation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, feb 2018.

[247] Paul Voigtlaender, Jonathon Luiten, and Bastian Leibe. BoLTVOS: Box-Level Tracking for Video Object Segmentation. apr 2019.

[248] F Perazzi, J Pont-Tuset, B. McWilliams, L. Van Gool, M Gross, and A Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[249] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS Challenge on Video Object Segmentation. apr 2017.

[250] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark. sep 2018.

[251] Mennatullah Siam, Boris Oreshkin, and Martin Jagersand. AMP: Adaptive masked proxies for few-shot segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, feb 2019.

[252] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.

[253] Pedro O. Pinheiro, Tsung Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*. Springer Verlag, mar 2016.

[254] Zongpu Zhang, Yang Hua, Tao Song, Zhengui Xue, Ruhui Ma, Neil Robertson, and Haibing Guan. Tracking-assisted weakly supervised online visual object segmentation in unconstrained videos. In *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, New York, New York, USA, oct 2018. Association for Computing Machinery, Inc.

[255] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5289–5298, June 2021.

[256] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. jun 2017.

[257] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, 2014.

[258] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 2010.

[259] Hang Qi, Matthew Brown, and David G. Lowe. Low-Shot Learning with Imprinted Weights. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[260] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations*, 2015.

[261] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, apr 2017.

[262] Jingchun Cheng, Yi Hsuan Tsai, Wei Chih Hung, Shengjin Wang, and Ming Hsuan Yang. Fast and Accurate Online Video Object Segmentation via Tracking Parts. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, dec 2018.

[263] Anjali Jaiprakash, William B O'Callaghan, Sarah L Whitehouse, Ajay Pandey, Liao Wu, Jonathan Roberts, and Ross W Crawford. Orthopaedic surgeon attitudes towards current limitations and the potential for robotic and technological innovation in arthroscopic surgery. *Journal of Orthopaedic Surgery*, 25(1):230949901668499, jan 2017.

[264] Liao Wu, Anjali Jaiprakash, Ajay K. Pandey, Davide Fontanarosa, Yaqub Jonmohamadi, Maria Antico, Mario Strydom, Andrew Razjigaev, Fumio Sasazawa, Jonathan Roberts, and Ross Crawford. Robotic and image-guided knee arthroscopy. In *Elsevier's Handbook of Robotic and Image-Guided Surgery*. 2018.

[265] Daniel R. Lueders, Jay Smith, and Jacob L. Sellon. Ultrasound-Guided Knee Procedures. *Physical Medicine and Rehabilitation Clinics of North America*, 27(3):631–648, aug 2016.

[266] Maria Antico, Fumio Sasazawa, Liao Wu, Anjali Jaiprakash, Jonathan Roberts, Ross Crawford, Ajay K. Pandey, and Davide Fontanarosa. Ultrasound guidance in minimally invasive robotic procedures. *Medical Image Analysis*, jan 2019.

[267] Y.S. Akgul, C. Kambhamettu, and M. Stone. Automatic extraction and tracking of the tongue contours. *IEEE Transactions on Medical Imaging*, 18(10):1035–1045, 1999.

[268] Anastasios Roussos, Athanassios Katsamanis, and Petros Maragos. Tongue tracking in Ultrasound images with Active Appearance Models. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1733–1736. IEEE, nov 2009.

[269] Gustavo Carneiro and Jacinto C. Nascimento. Combining multiple dynamic models and deep learning architectures for tracking the left ventricle endocardium in ultrasound data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2592–2607, nov 2013.

[270] Xiaojie Huang, Donald P Dione, Colin B Compas, Xenophon Papademetris, Ben A Lin, Alda Bregasi, Albert J Sinusas, Lawrence H Staib, and James S Duncan. Contour tracking in echocardiographic sequences via sparse representation and dictionary learning. *Medical image analysis*, 18(2):253–71, feb 2014.

[271] J. Guerrero, S.E. Salcudean, J.A. McEwen, B.A. Masri, and S. Nicolaou. Real-Time Vessel Segmentation and Tracking for Ultrasound Imaging Applications. *IEEE Transactions on Medical Imaging*, 26(8):1079–1090, aug 2007.

[272] V De Luca, T Benz, S Kondo, L König, D Lübke, S Rothlübbers, O Somphone, S Allaire, M A Lediju Bell, D Y F Chung, A Cifor, C Grozea, M Günther, J Jenne, T Kipshagen, M Kowarschik, N Navab, J Rühaak, J Schwaab, and C Tanner. The 2014 liver ultrasound tracking benchmark. *Physics in medicine and biology*, 60(14):5571–99, jul 2015.

[273] Alvaro Gomariz, Weiye Li, Ece Ozkan, Christine Tanner, and Orcun Goksel. Siamese Networks with Location Prior for Landmark Tracking in Liver Ultrasound Sequences. jan 2019.

[274] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[275] D. Nouri and A. Rothberg. Liver Ultrasound Tracking using a Learned Distance Metric. In *Proceedings of MICCAI workshop: Challenge on Liver Ultrasound Tracking*, pages 5–12, 2015.

[276] Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip H. S. Torr, and Andrea Vedaldi. *Learning feed-forward one-shot learners.* NEURAL INFO PROCESS SYS F, 2016.

[277] Lee R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302, jul 1945.

[278] T Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biol. Skr.*, 5:1–34, 1948.

[279] Jean Léger, Eliott Brion, Umair Javaid, John Lee, Christophe De Vleeschouwer, and Benoit Macq. Contour Propagation in CT Scans with Convolutional Neural Networks. In *Advanced Concepts for Intelligent Vision Systems*, pages 380–391, 2018.

[280] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese Instance Search for Tracking. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1420–1429. IEEE, jun 2016.

[281] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning Video Object Segmentation from Static Images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3491–3500. IEEE, jul 2017.

[282] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2141–2148. IEEE, jun 2010.

[283] David Tsai, Matthew Flagg, Atsushi Nakazawa, and James M. Rehg. Motion Coherent Tracking Using Multi-label MRF Optimization. *International Journal of Computer Vision*, 100(2):190–202, nov 2012.

[284] Nicolas Marki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. Bilateral Space Video Segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 743–751. IEEE, jun 2016.

[285] Paul Voigtlaender and Bastian Leibe. Online Adaptation of Convolutional Neural Networks for Video Object Segmentation. jun 2017.

[286] K.-K. Maninis, S Caelles, Y Chen, J Pont-Tuset, L Leal-Taixé, D Cremers, and L Van Gool. Video Object Segmentation Without Temporal Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[287] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. nov 2014.

[288] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, oct 2016.

[289] Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT Flow: Dense Correspondence across Scenes and Its Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, may 2011.

[290] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, aug 2013.

[291] Joseph Tighe and Svetlana Lazebnik. Finding Things: Image Parsing with Regions and Per-Exemplar Detectors. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3001–3008. IEEE, jun 2013.

[292] Pedro O Pinheiro and Ronan Collobert. Recurrent Convolutional Neural Networks for Scene Labeling. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–82—-I–90. JMLR.org, 2014.

[293] A Ben-Cohen, I Diamant, E Klang, M Amitai, and H Greenspan. Deep learning and data labeling for medical applications. In *Proceedings of the International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis. In: Lecture Notes in Computer Science*, volume 10008, pages 77–85, 2016.

[294] Ozan Oktay, Wenjia Bai, Matthew Lee, Ricardo Guerrero, Konstantinos Kamnitsas, Jose Caballero, Antonio de Marvao, Stuart Cook, Declan O'Regan, and Daniel Rueckert. Multi-input Cardiac Image Super-Resolution Using Convolutional Neural Networks. pages 246–254. 2016.

[295] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. pages 424–432. Springer, Cham, oct 2016.

[296] Lequan Yu, Xin Yang, Hao Chen, Jing Qin, and Pheng-Ann Heng. Volumetric ConvNets with mixed residual connections for automated prostate segmentation from 3D MR images, 2017.

[297] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. feb 2015.

[298] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[299] B. L. Welch. The Generalization Of Student's Problem When Several Different Population Variances Are Involved. *Biometrika*, 34(1-2):28–35, 01 1947.

[300] Benoit Steiner, Zachary Devito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, Alban Desmaison, Alykhan Tejani, Andreas Kopf, James Bradbury, Luca Antiga, Martin Raison, Natalia Gimelshein, Sasank Chilamkurthy, Trevor Killeen, Lu Fang, and Junjie Bai. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 2019.

[301] Roman Pflugfelder. An In-Depth Analysis of Visual Tracking with Siamese Neural Networks. jul 2017.

[302] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. nov 2015.